Econ821 Problem Set 1 Result Summaries

Xuan Lin

The data analyses for hedonic models were conducted in R. Codes are attached at the end of the file in PDF format. The original R file is uploaded on github.

Question 1

Report means and variances of attributes for each city:

city	county	price_mean	year_built_mean	sq_footage_mean	bathrooms_mean	bedrooms_mean	total_rooms_mean	stories_mean	violent_crime_rate_mean	property_crime_rate_mean	year_of_sale_mean
LA	37	311463.9263	1952.01876	1602.58218	1.98569178	3.09157726	7.93244377	1.13457183	618.80952	1976.515392	1999.91664
LA	59	296527.6326	1981.45209	1541.96509	2.15192731	2.58722467	6.10440529	1.59174009	314.4707	1414.156806	1999.93998
LA	65	181315.3171	1973.89449	1623.62299	2.20300657	3.12359778	6.13233957	1.19949469	616.38208	2436.70124	1999.82567
LA	71	188939.8284	1979.33519	1679.6764	2.14879856	3.23983488	6.5127451	1.36482972	567.66939	2216.85848	2000.39754
LA	111	332179.352	1978.34596	1825.54504	2.3424373	3.34221863	6.64299928	1.48824648	335.66362	1241.452924	2000.05462
SF	1	398873.1171	1962.17447	1672.36158	2.0314922	3.17236625	6.50623577	1.43417452	441.49934	1971.128292	2000.34516
SF	13	374582.3608	1976.76117	1833.1482	2.24779821	3.30335691	8.01458746	1.43224894	415.78327	1898.750281	2000.55541
SF	75	501817.0863	1945.13257	1497.93269	1.73122589	2.61824094	5.86750359	1.34167738	586.16285	2141.653726	1999.94978
SF	81	503715.0083	1967.16667	1587.40496	2.04414991	2.73966838	5.93399993	1.35557314	349.07914	1713.204285	2000.26109
SF	85	464683.1897	1971.45669	1662.78732	2.16185856	3.21450405	6.84787873	1.40198467	322.55811	1358.682926	2000.23945

city	county	price_var	year_built_var	sq_footage_var	bathrooms_var	bedrooms_var	total_rooms_var	stories_var	violent_crime_rate_var	property_crime_rate_var	year_of_sale_var
LA	37	33942721831	240.439066	388140.973	0.63658388	0.69428087	4.26628219	0.11935241	77212.929	363360.2422	17.6570308
LA	59	30953601267	61.1363763	582421.099	0.45335844	0.82331954	2.32547899	0.26958701	18926.972	206208.1516	15.4169279
LA	65	11065383445	212.658027	329746.33	0.4548194	0.63225427	1.86934169	0.16273662	77673.747	681390.2369	17.903409
LA	71	12133249510	329.251437	389720.428	0.38646142	0.70516969	2.65746045	0.23528148	61450.635	368819.3119	17.3121685
LA	111	29798698555	261.878758	563030.2	0.51802005	0.8037673	2.66677468	0.26150082	23602.373	112945.0221	17.0726339
SF	1	41544980702	697.85835	437987.58	0.57241768	0.7728215	2.46933555	0.26079434	39922.396	371046.5242	17.7719081
SF	13	46561031579	379.707879	563350.088	0.50037351	0.83612014	4.26637152	0.24541211	77245.919	359813.9069	17.1362193
SF	75	65620319073	966.905052	365112.226	0.65790174	1.02139467	3.49386373	0.31268353	43581.41	122186.3927	18.3596019
SF	81	73286582870	462.758853	466259.415	0.56517718	0.97725774	3.6362812	0.25108999	42814.557	1468429.518	17.8774106
SF	85	55932151423	347.083201	410670.707	0.44054718	0.8921065	3.48604034	0.27053424	17379.229	80485.56847	17.9127809

Question 2

points estimates and Bootstrapping results

```
Call:
boot(data = la_data, statistic = bs, R = 500, formula = model)


Bootstrap Statistics :
          original        bias     std. error
t1*   1.989657e+06  1.318247e+05 1.498994e+06 Constant
t2*   1.864903e+04  3.698016e+01 4.833379e+02 Bathroom
t3*  -2.493170e+04  1.235183e+00 3.798288e+02 Bedroom
t4*  -8.226933e+02  2.001854e+01 5.588499e+02 Stories
t5*   3.719187e+00 -9.710739e-02 1.439251e+00 PCR
t6*   4.648937e-04  1.670576e-05 2.359402e-04 PCR^2
t7*  -8.931034e+02 -1.342622e+02 1.528999e+03 Year_built
t8*  -2.926118e-02  3.421392e-02 3.899573e-01 Year_built^2
t9*   1.482554e+02 -1.494687e-02 2.396087e+00 Square_footage
t10* -4.293439e-05 -1.033138e-05 5.547963e-04 Square_footage^2
t11* -3.354241e+03 -5.874402e+00 7.265847e+02 Total Room
t12*  5.521781e+02  2.813446e-01 4.469041e+01 Total Room^2
t13* -2.022722e+02 -7.585901e-02 3.033704e+00 VCR
t14*  7.382382e-02  6.909663e-05 1.648867e-03 VCR^2
t15*  2.566848e+04 -3.930548e+01 9.007504e+02 year/county dummies
t16*  7.701509e+03  3.342162e+01 1.020723e+03
t17* -6.539264e+03 -2.279894e+00 9.695666e+02
t18* -1.364771e+04  2.527797e+01 8.746022e+02
t19* -1.464752e+04  2.306539e+01 8.820465e+02
t20* -5.470540e+03 -1.388764e+01 7.952660e+02
t21*  1.079181e+04 -1.459507e+01 7.770563e+02
t22*  2.463901e+04  1.320286e+01 7.722553e+02
t23*  4.889535e+04 -1.641579e+01 8.065821e+02
t24*  8.847804e+04 -5.150423e+01 8.735586e+02
t25*  1.498607e+05 -2.011449e+00 9.719851e+02
t26*  2.016449e+05  5.008323e+01 1.015590e+03
t27*  2.211715e+05 -3.594313e+01 1.024624e+03
t28*  2.062445e+05  1.253221e+01 1.304568e+03
t29*  7.796900e+04  2.817227e+01 1.186984e+03
t30* -1.239729e+04 -6.351995e+01 1.166373e+03
t31* -1.064795e+05 -1.513362e+01 8.184827e+02
t32* -1.111114e+05 -1.852039e+01 6.928437e+02
t33* -8.710331e+03 -7.827093e+01 7.906988e+02
```

```
Call:
boot(data = sf_data, statistic = bs, R = 500, formula = model_sf)


Bootstrap Statistics :
          original        bias     std. error
t1*  -2.107292e+07  2.440171e+03 1.597930e+06 Constant
t2*   2.168005e+04  2.386806e+01 6.933623e+02 Bathroom
t3*  -1.933357e+04  1.014811e+01 4.273925e+02 Bedroom
t4*  -2.840383e+04  3.365967e+01 5.910538e+02 Stories
t5*  -8.203774e+01  3.692447e-02 1.886203e+00 PCR
t6*   9.319009e-03 -1.154506e-06 2.886577e-04 PCR^2
t7*   2.308429e+04 -2.259161e+00 1.630007e+03 Year_built
t8*  -6.270300e+00  5.527835e-04 4.156644e-01 Year_built^2
t9*   2.347345e+02 -8.857729e-02 3.253872e+00 Square_footage
t10* -1.006847e-03  1.494209e-05 7.221151e-04 Square_footage^2
t11*  3.072683e+04 -1.927611e+01 1.138151e+03 Total Room
t12* -1.905501e+03  2.277221e-01 7.789796e+01 Total Room^2
t13* -2.251711e+02 -2.699683e-01 4.004284e+00 VCR
t14*  1.116099e-01  1.321667e-04 2.123464e-03 VCR^2
t15* -4.604047e+04  1.662539e+01 1.048760e+03 year/county dummies
t16* -1.740659e+04  2.982994e+01 1.131023e+03
t17* -3.250213e+04 -1.432047e+01 1.098623e+03
t18* -4.395174e+04 -1.354667e+01 1.058030e+03
t19* -2.290150e+04 -6.975050e+00 1.032970e+03
t20* -6.639792e+03 -1.430323e+01 1.004120e+03
t21*  6.569116e+04  8.390140e+01 1.089710e+03
t22*  8.078236e+04 -4.649580e+01 1.166894e+03
t23*  9.007342e+04 -7.612461e+01 1.141396e+03
t24*  1.083045e+05 -6.126087e+00 1.063366e+03
t25*  1.661588e+05 -1.384953e+01 1.171738e+03
t26*  2.555370e+05 -1.199936e+01 1.172382e+03
t27*  2.605482e+05 -9.132196e+01 1.304497e+03
t28*  2.405591e+05 -8.501107e+01 1.565719e+03
t29*  1.118781e+05 -9.948590e+01 1.795105e+03
t30* -5.835151e+04 -2.622206e+01 6.157978e+02
t31*  1.544596e+05 -7.072409e+01 1.762366e+03
t32*  9.209555e+04 -4.436857e+01 1.103500e+03
t33*  3.594948e+04 -3.365045e+01 5.996326e+02
```

Question 3

Point estimation by using the actual data:

The first stage: Coefficients of VCR and VCR SQUARE

- LA: -2.022722e+02; 7.382382e-02
- SF: -2.251711e+02 1.116099e-01

The Second Stage:

```
Call:
lm(formula = implicit_price ~ violent_crime_rate + buyers_data.3 +
    buyers_data.4 + buyers_data.5 + income + LA_indicator, data = buyers_data)

Residuals:
    Min      1Q  Median      3Q     Max
-65.336  -4.497   0.421   3.734 175.636

Coefficients:
                     Estimate Std. Error   t value Pr(>|t|)
(Intercept)        -2.033e+02  3.485e-02 -5834.460  < 2e-16 ***
violent_crime_rate  1.690e-01  4.265e-05  3962.643  < 2e-16 ***
buyers_data.3      -3.366e-01  5.840e-02    -5.765 8.19e-09 ***
buyers_data.4      -7.169e-01  3.550e-02   -20.192  < 2e-16 ***
buyers_data.5       1.507e-01  2.943e-02     5.123 3.01e-07 ***
income             -1.496e-06  9.392e-08   -15.925  < 2e-16 ***
LA_indicator       -9.840e+00  2.366e-02  -415.932  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.639 on 659541 degrees of freedom
Multiple R-squared:  0.9617,    Adjusted R-squared:  0.9617
F-statistic: 2.761e+06 on 6 and 659541 DF,  p-value: < 2.2e-16

>
> ######################standard eroors by taking sd of bootstrapped parameters:0.003297735
> secondstage_boot <-read.csv("second_stage_500.csv")
> sd(secondstage_boot$violent_crime_rate)
[1] 0.003297735
```

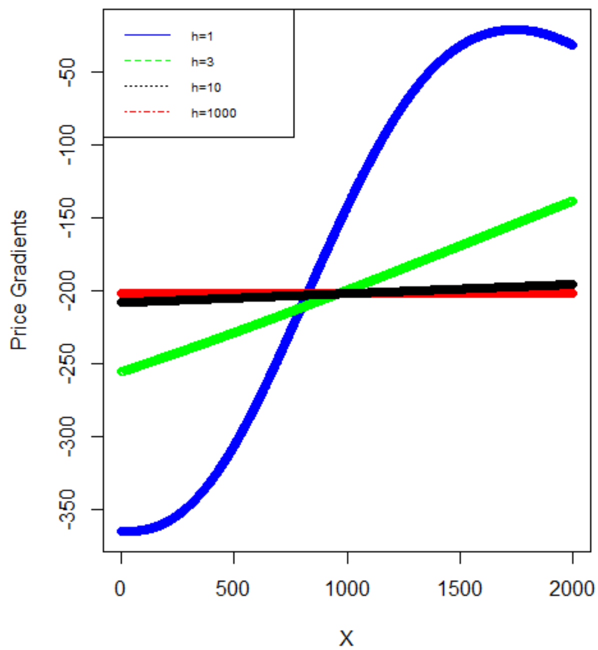
standard errors by taking sd of bootstrapped parameters

```
> secondstage_boot <-read.csv("second_stage_500.csv")
> sd(secondstage_boot$violent_crime_rate)
[1] 0.003297735
>
```

Analysis: The first stage results tell that housing price is negatively correlated with crime rate for both cities. (sensible) The second stage of Rosen illustrates that people's MWTP for violent(negative) is positively correlated with the violent level. People care less about the crime behaviours in the neighborhood when crime level is higher. They ask for less compensation in terms of housing price (that sounds weird to me) Possible explanation: people who cares less about safety lives in the higher-crime communities. Also, People with higher income care more about the crime. But the difference is tiny.

Question 4

Hedonic price gradients, non-parametric measurement:



Question 5

MWTP estimates based on Non-parametric gradient

```
> summary(MWTP_estimation)

Call:
lm(formula = MWTP ~ income + Asian_pi + black + hispanic, data = la_buyer_data)

Residuals:
    Min      1Q  Median      3Q     Max 
-117.12  -46.35  -22.99   22.67  401.42 

Coefficients:
              Estimate Std. Error  t value Pr(>|t|)    
(Intercept) -2.953e+02  4.684e-01 -630.425   <2e-16 ***
income      -3.363e-05  2.438e-06  -13.793   <2e-16 ***
Asian_pi    -7.623e-01  8.912e-01   -0.855    0.392    
black        5.082e+01  1.383e+00   36.753   <2e-16 ***
hispanic     2.148e+01  7.029e-01   30.552   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 68.67 on 55493 degrees of freedom
Multiple R-squared:  0.04225,	Adjusted R-squared:  0.04218 
F-statistic:   612 on 4 and 55493 DF,  p-value: < 2.2e-16
```

Question 6

Point estimation by using the actual data:

```
> theta.start = c(-200,0.169, 0,0,0,0)
> names(theta.start) = c("mu1", "mu2", "mu3", "mu4", "mu5", "mu6")
> theta.mle = optim(par=theta.start, fn=mle, x=buyers_data, method ="BFGS")
> theta.mle
$par     intercept       violent crime         income          asian_pi          black        hispanic
           mu1            mu2            mu3            mu4            mu5            mu6
-1.939636e+02  4.428857e-01  9.380051e+05  9.751060e-01  2.879410e-01  1.159335e+00

$value
[1] -18826365

$counts
function gradient
     145       29

$convergence
[1] 0

$message
NULL
```

Compared to estimation result in Q3, the coefficient of crime rate is higher in the MLE estimation. Coefficient of income seems not that sensible to me.

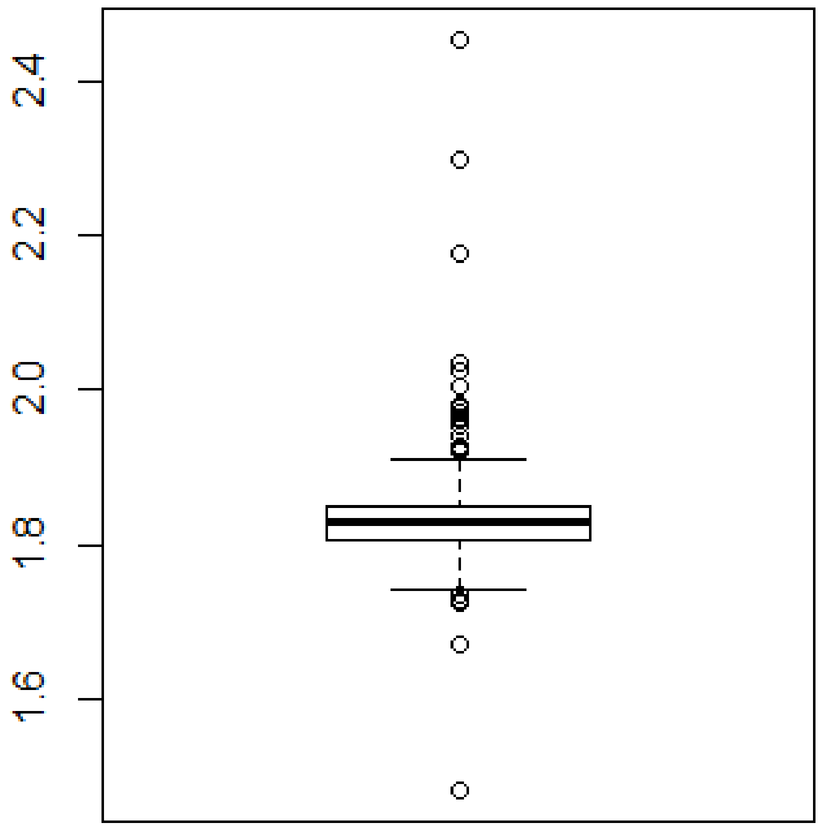
a few thoughts about the MLE method:

- MLE there is very sensitive to the starting point
- the likelihood may turn to be zero sometimes. We have to drop some observations to make the MLE method feasible. The number of drop out will influence the validity and reliability of the result.

Bootstrapping

```
> violent_crime
  [1] 1.782217 1.815562 1.827490 1.794456 1.880261 1.925557 1.889934 1.774643 1.858079 1.774413 1.833943 1.792788 1.801808 1.806541 1.824256 1.805515
 [17] 1.796021 1.923405 1.838094 1.856386 1.776214 1.804366 1.871373 1.827608 1.828061 1.812526 1.795154 1.851288 1.798909 1.821304 1.796239 1.862773
 [33] 1.833526 1.858454 1.899362 1.842968 1.809822 1.841034 1.790144 1.828057 1.814842 1.901679 1.831361 1.844445 1.853300 1.836709 1.818653 1.832467
 [49] 1.821470 1.815543 1.777509 1.798629 1.669396 1.798543 1.826587 1.829936 1.723054 1.824571 1.830146 1.852022 1.831964 1.849918 1.789620 1.750084
 [65] 1.836406 1.875767 1.835583 1.833205 1.805031 1.826747 1.827572 1.875013 1.786784 1.835285 1.863964 1.824027 1.850505 1.833228 1.830584 1.814005
 [81] 1.832528 1.844671 1.754401 1.742563 1.734933 1.857054 1.810795 1.733734 1.773632 1.847787 1.742937 1.801780 1.813951 1.955383 1.819451 1.823219
 [97] 1.808826 1.828213 1.801898 1.822880 1.829228 1.868184 2.455006 1.805743 1.858498 1.798693 1.806809 1.844918 1.782432 1.847236 1.804924 1.829262
[113] 1.845973 1.843796 1.830253 1.779808 1.816417 1.823904 1.831860 1.860693 1.841052 1.852699 1.911741 1.823685 1.791700 1.795314 1.842719 1.857360
[129] 1.741861 1.773655 1.829539 1.875210 1.881200 1.807225 1.834390 1.835673 1.835521 1.858576 1.821006 1.939656 1.928155 1.808461 1.847885 1.838589
[145] 1.802460 1.756794 1.831751 1.841636 1.845289 1.865521 1.819513 1.811447 1.856956 1.844261 1.982415 1.884124 1.803105 1.830127 1.817734 1.815931
[161] 1.824232 1.757649 1.838201 1.827747 1.829993 1.776675 1.818443 1.812491 1.810877 1.893814 1.830844 1.798633 1.801474 1.755423 1.875788 1.807063
[177] 1.826129 1.835223 1.981046 1.851005 1.800713 1.857467 1.823567 1.822516 1.792560 1.797511 1.828752 1.811588 1.787637 1.856371 1.814268 1.842550
[193] 1.961541 1.819967 1.867458 1.771259 1.787259 1.831562 1.816855 1.881897 1.832008 1.806375 1.845611 1.814113 1.810295 1.864097 1.828835 1.776685
[209] 1.881765 1.827459 1.895581 1.811705 1.963601 1.807496 1.810979 1.831902 1.870396 1.805486 1.851507 1.864688 1.823412 1.825653 1.809318 1.857794
[225] 1.823519 1.877464 1.860671 1.795439 1.879195 1.821495 1.846106 1.799992 1.813219 1.802394 1.882108 1.782057 1.843765 2.177457 1.823419 1.842632
[241] 1.814171 1.798920 1.866874 1.805572 1.781266 1.811242 1.871169 1.829878 1.856727 1.891387 1.847272 1.837774 1.799051 1.834240 1.847345 1.771250
[257] 1.878384 1.832366 1.820039 1.852922 1.844577 1.778527 1.828818 1.822788 1.848795 1.770491 1.841588 1.857984 1.831653 1.884310 1.750739 1.866524
[273] 1.834363 1.850884 1.822605 1.480867 1.783974 1.787437 1.834372 1.847052 2.034499 1.842540 1.839407 1.851192 1.782973 1.814291 1.771079 1.797482
[289] 1.847628 1.800188 1.981723 1.850050 1.817048 1.782921 1.783625 1.781117 1.782047 1.838412 1.814003 1.845232 1.812335 1.876023 1.840056 1.836015
[305] 1.919997 1.811184 1.919188 1.827879 1.839042 1.982109 1.825886 1.851687 1.792556 1.751103 1.808232 1.813620 1.828643 1.818329 1.818164 1.821381
[321] 1.826291 1.827385 1.976472 1.807699 1.828256 1.745108 1.827543 1.814002 1.825401 1.811616 1.819756 1.853961 1.831233 1.787733 1.819050 1.811777
[337] 1.837244 1.828785 1.851768 1.776763 1.820340 1.857451 1.824107 1.826305 1.849786 1.846581 1.800854 1.830895 1.806507 1.813493 1.870385 1.838442
[353] 1.856770 1.804363 1.876558 1.858419 1.794771 1.971303 1.837848 1.850002 1.760206 1.869202 1.787074 1.811532 1.804852 1.870076 1.759275 1.816105
[369] 1.846268 1.797690 1.886622 1.810495 1.814965 1.838999 1.862691 1.802184 1.843593 1.802868 1.806763 1.864329 1.821768 1.798859 1.848888 1.785530
[385] 1.845353 1.787477 1.799120 1.858385 1.841436 1.797106 1.812691 1.834111 1.802344 1.842712 1.876698 1.778594 1.884942 1.963915 1.832281 1.835261
[401] 1.793340 1.820506 1.854267 1.850917 1.799769 1.825448 1.842333 1.788263 1.802643 1.780569 1.867966 1.784078 1.834533 1.820409 1.836481 1.775132
[417] 1.908003 1.844489 1.815312 2.023438 1.822324 1.851950 1.817398 1.801072 1.919310 1.797241 1.763500 1.768706 1.795012 1.780384 1.875702 1.795257
[433] 1.873304 1.785825 1.804800 1.830827 1.747224 1.843068 1.832453 1.864628 1.782407 1.805847 1.856219 1.843970 1.838836 1.815265 1.804331 1.881960
[449] 2.300400 1.833541 1.786054 1.778829 1.843361 1.788356 1.799483 1.843943 1.827247 1.840947 1.781895 1.894778 1.833538 1.769212 1.845419 1.826205
[465] 1.723023 1.847063 1.803830 1.831288 1.827754 1.804388 1.871295 1.855843 1.853210 1.901544 1.875129 1.862622 1.861755 1.851906 1.726842 1.845828
[481] 1.851554 1.846555 1.795624 1.818731 1.814058 1.803125 1.833952 1.773778 1.850740 1.831377 1.823737 1.816884 1.824068 2.005891 1.813043 1.851244
[497] 1.815051
> mean(violent_crime)
[1] 1.831635
> sd(violent_crime)
[1] 0.05983717
```

three estimations were removed as outliers before the calculation of mean and standard deviation



boxplot for the coefficient of violent-crime-rate on the "second stage" MLE.

```r
setwd("/Volumes/USB30FD/821_ps/hedonic")
library(dplyr)
library(purrr)
library(psych)
#Q1
############################################################
la_data<- read.table("la_data.txt", header = FALSE)
names(la_data) <-
c("houseid","price","county","year_built","sq_footage","bathrooms","bedrooms","total_roo
"year_of_sale")
la_sta <- la_data %>%
  group_by(county) %>%
  summarise(price_mean=mean(price), year_built_mean=mean(year_built),
sq_footage_mean=mean(sq_footage),
bathrooms_mean=mean(bathrooms),bedrooms_mean=mean(bedrooms),
total_rooms_mean=mean(total_rooms),stories_mean=mean(stories),
violent_crime_rate_mean=mean(violent_crime_rate),
property_crime_rate_mean=mean(property_crime_rate),
year_of_sale_mean=mean(year_of_sale),
            price_var=var(price), year_built_var=var(year_built),
sq_footage_var=var(sq_footage),
bathrooms_var=var(bathrooms),bedrooms_var=var(bedrooms),
total_rooms_var=var(total_rooms),stories_var=var(stories),
violent_crime_rate_var=var(violent_crime_rate),
property_crime_rate_var=var(property_crime_rate),
year_of_sale_var=var(year_of_sale))
la_sta$city <- "LA"

sf_data<- read.table("sf_data.txt", header = FALSE)
names(sf_data) <-
c("houseid","price","county","year_built","sq_footage","bathrooms","bedrooms","total_roo
"year_of_sale")
sf_sta <-sf_data %>%
  subset(., sf_data$county==1|13|75|81|85) %>%
  group_by(county) %>%
  summarise(price_mean=mean(price), year_built_mean=mean(year_built),
sq_footage_mean=mean(sq_footage),
bathrooms_mean=mean(bathrooms),bedrooms_mean=mean(bedrooms),
total_rooms_mean=mean(total_rooms),stories_mean=mean(stories),
violent_crime_rate_mean=mean(violent_crime_rate),
property_crime_rate_mean=mean(property_crime_rate),
year_of_sale_mean=mean(year_of_sale),
            price_var=var(price), year_built_var=var(year_built),
sq_footage_var=var(sq_footage),
bathrooms_var=var(bathrooms),bedrooms_var=var(bedrooms),
total_rooms_var=var(total_rooms),stories_var=var(stories),
violent_crime_rate_var=var(violent_crime_rate),
property_crime_rate_var=var(property_crime_rate),
year_of_sale_var=var(year_of_sale))
sf_sta$city <- "SF"

##combine two stats table
total_sta <- rbind(la_sta, sf_sta)
write.csv(total_sta, file="Q1_result.csv")
```

```
##########################################################################

#Q2

rm(list=ls())
######################################################
installation_needed  <- TRUE
loading_needed <- TRUE
package_list <- c('foreign', 'xtable', 'plm','gmm',
'AER','stargazer','readstata13', 'boot')
if(installation_needed){install.packages(package_list, repos='http://
cran.us.r-project.org')}
if(loading_needed){lapply(package_list, require, character.only = TRUE)}
library(boot)
library("dummies")
library(dplyr)
library(purrr)
library(psych)
##########################
# data & model_LA
la_data<- read.table("la_data.txt", header = FALSE)
names(la_data) <-
c("houseid","price","county","year_built","sq_footage","bathrooms","bedrooms","total_roo
"year_of_sale")
la_data <- cbind(la_data, dummy(la_data$county, sep = "."))
la_data <- cbind(la_data, dummy(la_data$year_of_sale, sep = "."))
model_la <- price ~ bathrooms + bedrooms + stories + property_crime_rate +
I(property_crime_rate^2) + year_built + I(year_built^2) + sq_footage +
I(sq_footage^2) + total_rooms + I(total_rooms^2) + violent_crime_rate +
I(violent_crime_rate^2) + bathrooms + la_data.1993 + la_data.1994 +
la_data.1995 + la_data.1996 + la_data.1997 + la_data.1998 + la_data.2000 +
la_data.2001 + la_data.2002 + la_data.2003 + la_data.2004 + la_data.2005 +
la_data.2006 + la_data.2007 + la_data.2008 + la_data.59 + la_data.65 +
la_data.71 + la_data.111
# function to obtain regression weights
bs <- function(formula, data, indices) {
  d <- data[indices,] # allows boot to select sample
  fit <- lm(formula, data=d)
  return(coef(fit))
}
# bootstrapping with 500 replications
results <- boot(data=la_data, statistic=bs,
                R=500, formula=model_la)
results
result_bootcef_la <- cbind(results$t[,13],results$t[,14])
write.csv(result_bootcef_la, file="result_bootcef_la.csv")
capture.output(results, file = "results_boots_la.txt", append = TRUE)
###################
# data & model_SF
sf_data<- read.table("sf_data.txt", header = FALSE)
names(sf_data) <-
c("houseid","price","county","year_built","sq_footage","bathrooms","bedrooms","total_roo
"year_of_sale")
select(sf_data, sf_data$county==1|13|75|81|85)
```

```
sf_data <- cbind(sf_data, dummy(sf_data$county, sep = "."))
sf_data <- cbind(sf_data, dummy(sf_data$year_of_sale, sep = "."))
model_sf <- price ~ bathrooms + bedrooms + stories + property_crime_rate +
I(property_crime_rate^2) + year_built + I(year_built^2) + sq_footage +
I(sq_footage^2) + total_rooms + I(total_rooms^2) + violent_crime_rate +
I(violent_crime_rate^2) + bathrooms + sf_data.1993 + sf_data.1994 +
sf_data.1995 + sf_data.1996 + sf_data.1997 + sf_data.1998 + sf_data.2000 +
sf_data.2001 + sf_data.2002 + sf_data.2003 + sf_data.2004 + sf_data.2005 +
sf_data.2006 + sf_data.2007 + sf_data.2008 + sf_data.13 + sf_data.75 +
sf_data.81 + sf_data.85

# bootstrapping with 500 replications
results_boot_sf <- boot(data=sf_data, statistic=bs,
                R=500, formula=model_sf)
results_boot_sf
result_bootcef_sf <- cbind(results_boot_sf$t[,13],results$t[,14])
write.csv(result_bootcef_sf, file="result_bootcef_sf.csv")
capture.output(results_boot_sf, file = "results_boots_sf.txt", append =
TRUE)
#####################################################################################

#Q3
rm(list=ls())
###################### data read-in
##buyers
buyers_data<- read.table("buyer_data_sf_la.txt", header = FALSE)
names(buyers_data) <-
c("buyerid","price","violent_crime_rate","property_crime_rate", "race",
"income", "LA_indicator")

## boostrapped hedonic price gradients from Q2
la_gradients <- read.csv("result_bootcef_la.csv")
sf_gradients <- read.csv("result_bootcef_sf.csv")
gradients_la_sf <- merge(la_gradients,sf_gradients, by="boot_round")
rm(la_gradients,sf_gradients)
######################assign implicit price of crime for each individual
*500
for( i in 1:500){
  buyers_data[paste("implict_price", i, sep= "")] <-
(buyers_data$LA_indicator)*gradients_la_sf[i,2]+ (1-
buyers_data$LA_indicator)*gradients_la_sf[i,4] + 2*gradients_la_sf[i,
3]*(buyers_data$violent_crime_rate)*(buyers_data$LA_indicator) +
2*gradients_la_sf[i,5]*(buyers_data$violent_crime_rate)*(1-
buyers_data$LA_indicator)
}
######################run second stage regression $ save coefficients *500
##create dummies for race
buyers_data <- cbind(buyers_data, dummy(buyers_data$race, sep = "."))

##regression
coef1 <-c()
for (j in 1:500){
  test_data <- cbind(buyers_data[, 1:7], buyers_data[, 508:511],
buyers_data[, 7+j])
```

```
  colnames(test_data)[12] <- "implicit_price"
  model <- lm(implicit_price ~ violent_crime_rate + buyers_data.3 +
buyers_data.4 + buyers_data.5 + income + LA_indicator, data=test_data)
  newcoef1 <- model$coef
  coef1 <- rbind(coef1,newcoef1)
}
print(coef1)
#write.csv(coef1, file="second_stage_500.csv")

######################the actural point estimation
##first stage of LA, result: -2.022722e+02; 7.382382e-02
la_data<- read.table("la_data.txt", header = FALSE)
names(la_data) <-
c("houseid","price","county","year_built","sq_footage","bathrooms","bedrooms","total_roo
"year_of_sale")
la_data <- cbind(la_data, dummy(la_data$county, sep = "."))
la_data <- cbind(la_data, dummy(la_data$year_of_sale, sep = "."))
model_la_actuall <- lm(price ~ bathrooms + bedrooms + stories +
property_crime_rate + I(property_crime_rate^2) + year_built +
I(year_built^2) + sq_footage + I(sq_footage^2) + total_rooms +
I(total_rooms^2) + violent_crime_rate + I(violent_crime_rate^2) + bathrooms
+ la_data.1993 + la_data.1994 + la_data.1995 + la_data.1996 + la_data.1997
+ la_data.1998 + la_data.2000 + la_data.2001 + la_data.2002 + la_data.2003
+ la_data.2004 + la_data.2005 + la_data.2006 + la_data.2007 + la_data.2008
+ la_data.59 + la_data.65 + la_data.71 + la_data.111, data=la_data)
summary(model_la_actuall)

##first stage of SF, result:  -2.251711e+02 1.116099e-01
sf_data<- read.table("sf_data.txt", header = FALSE)
names(sf_data) <-
c("houseid","price","county","year_built","sq_footage","bathrooms","bedrooms","total_roo
"year_of_sale")
select(sf_data, sf_data$county==1|13|75|81|85)
sf_data <- cbind(sf_data, dummy(sf_data$county, sep = "."))
sf_data <- cbind(sf_data, dummy(sf_data$year_of_sale, sep = "."))
model_sf_actuall <- lm(price ~ bathrooms + bedrooms + stories +
property_crime_rate + I(property_crime_rate^2) + year_built +
I(year_built^2) + sq_footage + I(sq_footage^2) + total_rooms +
I(total_rooms^2) + violent_crime_rate + I(violent_crime_rate^2) + bathrooms
+ sf_data.1993 + sf_data.1994 + sf_data.1995 + sf_data.1996 + sf_data.1997
+ sf_data.1998 + sf_data.2000 + sf_data.2001 + sf_data.2002 + sf_data.2003
+ sf_data.2004 + sf_data.2005 + sf_data.2006 + sf_data.2007 + sf_data.2008
+ sf_data.13 + sf_data.75 + sf_data.81 + sf_data.85, data=sf_data)
summary(model_sf_actuall)

##second stage
buyers_data<- read.table("buyer_data_sf_la.txt", header = FALSE)
names(buyers_data) <-
c("buyerid","price","violent_crime_rate","property_crime_rate", "race",
"income", "LA_indicator")
```

```
buyers_data$implicit_price <- (buyers_data$LA_indicator)*-2.022722e+02+ (1-
buyers_data$LA_indicator)*-2.251711e+02 +
2*7.382382e-02*(buyers_data$violent_crime_rate)*(buyers_data$LA_indicator)
+ 2*1.116099e-01*(buyers_data$violent_crime_rate)*(1-
buyers_data$LA_indicator)
buyers_data <- cbind(buyers_data, dummy(buyers_data$race, sep = "."))
model_second_stage_actuall <- lm(implicit_price ~ violent_crime_rate +
buyers_data.3 + buyers_data.4 + buyers_data.5 + income + LA_indicator,
data=buyers_data)
summary(model_second_stage_actuall)

########################standard eroors by taking sd of bootstrapped
parameters:0.003297735
secondstage_boot <-read.csv("second_stage_500.csv")
sd(secondstage_boot$violent_crime_rate)
###########################################################################################

#Q4
rm(list=ls())

library(dplyr)
library(purrr)
library(psych)
library(dummies)

##import data of LA
la_data<- read.table("la_data.txt", header = FALSE)
names(la_data) <-
c("houseid","price","county","year_built","sq_footage","bathrooms","bedrooms","total_roo
"year_of_sale")

##create a function that generates weight vectors (length: # observations)
for each X 1:2000 and h:1:3
la_data$weight <- 0
weight <-  function(x,h,theta){
  sigma <- sd(x[, 10])
  x[,13]<- (1/(h*sigma))*(1/sqrt(2*pi))*exp(-0.5*(((x[,10]-theta)/
(h*sigma))^2))
  ans <- x[,13]
  return(ans)
}

## for h=1
weight1 <- list()
for (i in 1:2000){
  weight1[[i]]<- weight(la_data, 1, i)
}
## for h=3
weight3 <- list()
for (i in 1:2000){
  weight3[[i]]<- weight(la_data, 3, i)
}
## for h=10
weight10 <- list()
```

```
for (i in 1:2000){
  weight10[[i]]<- weight(la_data, 10, i)
}

## for h=1000
weight1000 <- list()
for (i in 1:2000){
  weight1000[[i]]<- weight(la_data, 1000, i)
}

###############save gradient vector
##model
model <- price ~ violent_crime_rate
##h1
price_gradient_h1 <- replicate(2000,0)
for (i in 1:2000){
  estimation <-lm(model, la_data, weights=weight1[[i]])
  price_gradient_h1[i] <- estimation$coefficients[2]
}
save(price_gradient_h1, file = "h1_price_gradients.Rdata")
##h3
price_gradient_h3 <- replicate(2000,0)
for (i in 1:2000){
  estimation <-lm(model, la_data, weights=weight3[[i]])
  price_gradient_h3[i] <- estimation$coefficients[2]
}

##h10
price_gradient_h10 <- replicate(2000,0)
for (i in 1:2000){
  estimation <-lm(model, la_data, weights=weight10[[i]])
  price_gradient_h10[i] <- estimation$coefficients[2]
}

##h1000
price_gradient_h1000 <- replicate(2000,0)
for (i in 1:2000){
  estimation <-lm(model, la_data, weights=weight1000[[i]])
  price_gradient_h1000[i] <- estimation$coefficients[2] }
price_gradient_h1000

### We eventually get the final result! Let plot it now
x_axis <- c(1:2000)
plot (x_axis, price_gradient_h1, type="b", pch = 19, col="blue",
ylab="Price Gradients", xlab="X")
lines(x_axis, price_gradient_h1000, col="red", type="b")
lines(x_axis, price_gradient_h3, col="green", type="b")
lines(x_axis, price_gradient_h10, col="black", type="b")
legend("topleft", legend=c("h=1", "h=3","h=10", "h=1000"),
       col=c("blue", "green", "black", "red"), lty=1:4, cex=0.6)
#####################################################################################
###  Q5
rm(list=ls())
```

```
library(dplyr)
library(purrr)
library(psych)
library(dummies)

##import buyer data
la_buyer_data<- read.table("buyer_data_la.txt", header = FALSE)
names(la_buyer_data) <-
c("buyer_id","price","violent_crime_rate","property_crime_rate","race","income")
la_buyer_data <- cbind(la_buyer_data, dummy(la_buyer_data$race, sep = "."))
names(la_buyer_data) <-
c("buyer_id","price","violent_crime_rate","property_crime_rate","race","income",
                          "Asian_pi", "black", "hispanic", "white")

## import price gradients results
price_gradient <- readRDS("h1_price_gradients.Rdata")

##allocate WMTP to each individual
la_buyer_data$violent_crime_rate[la_buyer_data$violent_crime_rate >= 2000]
<- 2000
la_buyer_data$MWTP <-
price_gradient_h1[as.integer(la_buyer_data$violent_crime_rate)]
MWTP_estimation <- lm(MWTP~income+Asian_pi+black+hispanic,
data=la_buyer_data)
summary(MWTP_estimation)
#######################################################################################



###Q6
rm(list=ls())
#First_step:boostrap
######################################################
installation_needed  <- TRUE
loading_needed <- TRUE
package_list <- c('foreign', 'xtable', 'plm','gmm',
'AER','stargazer','readstata13', 'boot')
if(installation_needed){install.packages(package_list, repos='http://
cran.us.r-project.org')}
if(loading_needed){lapply(package_list, require, character.only = TRUE)}
library(boot)
library("dummies")
library(dplyr)
library(purrr)
library(psych)
##########################
# data & model_LA
la_data<- read.table("la_data.txt", header = FALSE)
names(la_data) <-
c("houseid","price","county","year_built","sq_footage","bathrooms","bedrooms","total_roo
"year_of_sale")
la_data <- cbind(la_data, dummy(la_data$county, sep = "."))
la_data <- cbind(la_data, dummy(la_data$year_of_sale, sep = "."))
```

```
model_la <- price ~ bathrooms + bedrooms + stories + property_crime_rate +
I(property_crime_rate^2) + year_built + I(year_built^2) + sq_footage +
I(sq_footage^2) + total_rooms + I(total_rooms^2)+ violent_crime_rate +
I(violent_crime_rate^2) + I(violent_crime_rate^3)+ I(violent_crime_rate^4)+
I(violent_crime_rate^5)+ I(violent_crime_rate^6) + bathrooms + la_data.1993
+ la_data.1994 + la_data.1995 + la_data.1996 + la_data.1997 + la_data.1998
+ la_data.2000 + la_data.2001 + la_data.2002 + la_data.2003 + la_data.2004
+ la_data.2005 + la_data.2006 + la_data.2007 + la_data.2008 + la_data.59 +
la_data.65 + la_data.71 + la_data.111

# function to obtain regression weights
bs <- function(formula, data, indices) {
  d <- data[indices,] # allows boot to select sample
  fit <- lm(formula, data=d)
  return(coef(fit))
}
# bootstrapping with 500 replications
results <- boot(data=la_data, statistic=bs,
                R=500, formula=model_la)
results
result_bootcef_la <- cbind(results$t[,13],results$t[,14],results$t[,
15],results$t[,16],results$t[,17],results$t[,18])
write.csv(result_bootcef_la, file="q6_result_bootcef_la.csv")
capture.output(results, file = "results_boots_la.txt", append = TRUE)
###################
# data & model_SF
sf_data<- read.table("sf_data.txt", header = FALSE)
names(sf_data) <-
c("houseid","price","county","year_built","sq_footage","bathrooms","bedrooms","total_roo
"year_of_sale")
select(sf_data, sf_data$county==1|13|75|81|85)
sf_data <- cbind(sf_data, dummy(sf_data$county, sep = "."))
sf_data <- cbind(sf_data, dummy(sf_data$year_of_sale, sep = "."))
model_sf <- price ~ bathrooms + bedrooms + stories + property_crime_rate +
I(property_crime_rate^2) + year_built + I(year_built^2) + sq_footage +
I(sq_footage^2) + total_rooms + I(total_rooms^2) + violent_crime_rate +
I(violent_crime_rate^2) + I(violent_crime_rate^3)+ I(violent_crime_rate^4)+
I(violent_crime_rate^5)+ I(violent_crime_rate^6) + bathrooms + sf_data.1993
+ sf_data.1994 + sf_data.1995 + sf_data.1996 + sf_data.1997 + sf_data.1998
+ sf_data.2000 + sf_data.2001 + sf_data.2002 + sf_data.2003 + sf_data.2004
+ sf_data.2005 + sf_data.2006 + sf_data.2007 + sf_data.2008 + sf_data.13 +
sf_data.75 + sf_data.81 + sf_data.85

# bootstrapping with 500 replications
results_boot_sf <- boot(data=sf_data, statistic=bs,
                        R=500, formula=model_sf)
results_boot_sf
result_bootcef_sf <- cbind(results_boot_sf$t[,13],results_boot_sf$t[,
14],results_boot_sf$t[,15],results_boot_sf$t[,16],results_boot_sf$t[,
17],results_boot_sf$t[,18])
write.csv(result_bootcef_sf, file="q6_result_bootcef_sf.csv")
capture.output(results_boot_sf, file = "results_boots_sf.txt", append =
TRUE)
#####################################################################################
```

```
##Bishop Timmins
library(mlogit)
library(dplyr)
library(maxLik)
##buyers data
buyers_data<- read.table("buyer_data_sf_la.txt", header = FALSE)
names(buyers_data) <-
c("buyerid","price","violent_crime_rate","property_crime_rate", "race",
"income", "LA_indicator")

## boostrapped hedonic price gradients from Q5
la_gradients <- read.csv("q6_result_bootcef_la.csv")
sf_gradients <- read.csv("q6_result_bootcef_sf.csv")
gradients_la_sf <- merge(la_gradients,sf_gradients, by="X")
rm(la_gradients,sf_gradients)

#####point estimation:
point_la <-lm(model_la, data=la_data)
point_gradient_la <- point_la$coefficients[13:18]
point_sf <- lm(model_sf, data=sf_data)
point_gradient_sf <- point_sf$coefficients[13:18]
point_gradient <- cbind(point_gradient_la, point_gradient_sf)
buyers_data$implicit_price <- (buyers_data$LA_indicator)*point_gradient[1,1]+ (1-
buyers_data$LA_indicator)*point_gradient[1,2]+
2*point_gradient[2,1]*(buyers_data$violent_crime_rate)*(buyers_data$LA_indicator) +
2*point_gradient[2,2]*(buyers_data$violent_crime_rate)*(1-buyers_data$LA_indicator)
+
3*point_gradient[3,1]*(buyers_data$violent_crime_rate^2)*(buyers_data$LA_indicator)
+ 3*point_gradient[3,2]*(buyers_data$violent_crime_rate^2)*(1-
buyers_data$LA_indicator)+
4*point_gradient[4,1]*(buyers_data$violent_crime_rate^3)*(buyers_data$LA_indicator)
+ 4*point_gradient[4,2]*(buyers_data$violent_crime_rate^3)*(1-
buyers_data$LA_indicator)+
5*point_gradient[5,1]*(buyers_data$violent_crime_rate^4)*(buyers_data$LA_indicator)
+ 5*point_gradient[5,2]*(buyers_data$violent_crime_rate^4)*(1-
buyers_data$LA_indicator)+
6*point_gradient[6,1]*(buyers_data$violent_crime_rate^5)*(buyers_data$LA_indicator)
+ 6*point_gradient[6,2]*(buyers_data$violent_crime_rate^5)*(1-
buyers_data$LA_indicator)
```

```
buyers_data$df_implict_price <- 2*point_gradient[2,1]*(buyers_data$LA_indicator) +
2*point_gradient[2,2]*(1-buyers_data$LA_indicator)+
6*point_gradient[3,1]*(buyers_data$violent_crime_rate)*(buyers_data$LA_indicator) +
6*point_gradient[3,2]*(buyers_data$violent_crime_rate)*(1-buyers_data$LA_indicator)+
12*point_gradient[4,1]*(buyers_data$violent_crime_rate^2)*(buyers_data$LA_indicator)
+ 12*point_gradient[4,2]*(buyers_data$violent_crime_rate^2)*(1-
buyers_data$LA_indicator)+
20*point_gradient[5,1]*(buyers_data$violent_crime_rate^3)*(buyers_data$LA_indicator)
+ 20*point_gradient[5,2]*(buyers_data$violent_crime_rate^3)*(1-
buyers_data$LA_indicator)+
30*point_gradient[6,1]*(buyers_data$violent_crime_rate^4)*(buyers_data$LA_indicator)
+ 30*point_gradient[6,2]*(buyers_data$violent_crime_rate^4)*(1-
buyers_data$LA_indicator)



##create dummies for race
buyers_data <- cbind(buyers_data, dummy(buyers_data$race, sep = "."))

#generate mle function
mle = function(theta, x) {
  # theta parameter vector; x
  mu1 = theta[1]
  mu2 = theta[2]
  mu3 = theta[3]
  mu4 = theta[4]
  mu5 = theta[5]
  mu6 = theta[6]
  #x[,14]:vij
  x[, 14] <- x[,8] - mu1 -mu2*x[,3]-mu3*x[,6]-mu4*x[,10]-mu5*x[,11]-mu6*x[,
12]
  sigma <- sd(x[,14])
  x[, 15] <- (1/(sigma*sqrt(2*pi)))*exp(-(1/(2*(sigma^2)))*(x[,
14]^2))*abs((x[,9]-mu2))
  x <- subset(x, x[,15] >0 ) ###We need to drop 0 value!!!!! fairly
important for mle method to work
  ans <- sum(log(x[,15]))
  return(ans)
}
#use ols to try the start point
try <- lm(implicit_price~ violent_crime_rate + income + buyers_data.2 +
buyers_data.3 + buyers_data.4, data=buyers_data)
theta.start = c(-200,0.169, 0,0,0,0)
names(theta.start) = c("mu1", "mu2", "mu3", "mu4", "mu5", "mu6")
#theta.mle = maxLik (mle, start=theta.start, x=buyers_data, method =
"BFGS")  ###the problem here is that mle could be very sensitive about the
start point
theta.mle = optim(par=theta.start, fn=mle, x=buyers_data, method ="BFGS")

######################assign df(implicit price) of crime for each
individual *500
buyers_data<- read.table("buyer_data_sf_la.txt", header = FALSE)
```

```
names(buyers_data) <-
c("buyerid","price","violent_crime_rate","property_crime_rate", "race",
"income", "LA_indicator")
for( i in 1:500){
  buyers_data[paste("implict_price", i, sep= "")] <-
(buyers_data$LA_indicator)*gradients_la_sf[i,2]+ (1-
buyers_data$LA_indicator)*gradients_la_sf[i,8]+ 2*gradients_la_sf[i,
3]*(buyers_data$violent_crime_rate)*(buyers_data$LA_indicator) +
2*gradients_la_sf[i,9]*(buyers_data$violent_crime_rate)*(1-
buyers_data$LA_indicator)+ 3*gradients_la_sf[i,
4]*(buyers_data$violent_crime_rate^2)*(buyers_data$LA_indicator) +
3*gradients_la_sf[i,10]*(buyers_data$violent_crime_rate^2)*(1-
buyers_data$LA_indicator)+ 4*gradients_la_sf[i,
4]*(buyers_data$violent_crime_rate^3)*(buyers_data$LA_indicator) +
4*gradients_la_sf[i,11]*(buyers_data$violent_crime_rate^3)*(1-
buyers_data$LA_indicator)+ 5*gradients_la_sf[i,
6]*(buyers_data$violent_crime_rate^4)*(buyers_data$LA_indicator) +
5*gradients_la_sf[i,12]*(buyers_data$violent_crime_rate^4)*(1-
buyers_data$LA_indicator)+ 6*gradients_la_sf[i,
7]*(buyers_data$violent_crime_rate^5)*(buyers_data$LA_indicator) +
6*gradients_la_sf[i,13]*(buyers_data$violent_crime_rate^5)*(1-
buyers_data$LA_indicator)
}
######################assign df(implicit price) of crime for each
individual *500
for( i in 1:500){
  buyers_data[paste("df_implict_price", i, sep= "")] <-
2*gradients_la_sf[i,3]*(buyers_data$LA_indicator) + 2*gradients_la_sf[i,
9]*(1-buyers_data$LA_indicator)+ 6*gradients_la_sf[i,
4]*(buyers_data$violent_crime_rate)*(buyers_data$LA_indicator) +
6*gradients_la_sf[i,10]*(buyers_data$violent_crime_rate)*(1-
buyers_data$LA_indicator)+ 12*gradients_la_sf[i,
4]*(buyers_data$violent_crime_rate^2)*(buyers_data$LA_indicator) +
12*gradients_la_sf[i,11]*(buyers_data$violent_crime_rate^2)*(1-
buyers_data$LA_indicator)+ 20*gradients_la_sf[i,
6]*(buyers_data$violent_crime_rate^3)*(buyers_data$LA_indicator) +
20*gradients_la_sf[i,12]*(buyers_data$violent_crime_rate^3)*(1-
buyers_data$LA_indicator)+ 30*gradients_la_sf[i,
7]*(buyers_data$violent_crime_rate^4)*(buyers_data$LA_indicator) +
30*gradients_la_sf[i,13]*(buyers_data$violent_crime_rate^4)*(1-
buyers_data$LA_indicator)
}

######################run second stage regression $ save coefficients *500
##create dummies for race
buyers_data <- cbind(buyers_data, dummy(buyers_data$race, sep = "."))
theta.start <- c(-1.939636e+02,  4.428857e-01,  9.380051e+05,
9.751060e-01,  2.879410e-01,  1.159335e+00)
names(theta.start) = c("mu1", "mu2", "mu3", "mu4", "mu5", "mu6")
intercept <- c()
violent_crime <- c()
income <- c()
asian_pi <-c()
black <- c()
```

```
hispanic <- c()
for (i in 1:500){
  sample <- buyers_data[, c(1:7,7+i,507+i,1008:1011)]
  theta.mle = optim(par=theta.start, fn=mle, x=sample, method ="BFGS")
  intercept[i] <- theta.mle$par[1]
  violent_crime[i] <- theta.mle$par[2]
  income[i] <- theta.mle$par[3]
  asian_pi[i] <- theta.mle$par[4]
  black[i] <- theta.mle$par[5]
  hispanic[i] <- theta.mle$par[6]
}

###drop outlier before summarizing boostrap results
install.packages("outliers")
library(outliers)
boxplot(income)
violent_crime <- rm.outlier(violent_crime)
violent_crime <- rm.outlier(violent_crime)

mean(violent_crime)
sd(violent_crime)
```

```
install.packages("mlogit")
install.packages("dplyr")
library(mlogit)
library(dplyr)
setwd("/Volumes/USB30FD/821_ps")
#####data upload and prepare
data <- read.csv("long_data.csv")
data <- data[-c(1)]
names(data)[names(data) == "idcase"] <- "id"
data <- mlogit.data(data, choice = "choice",
                    shape = "long", alt.levels = (c(1:100)), id ="id")
#Q2.i:try the simple model first
model_1 <- mlogit (choice ~ ramp + restroom + walleye + salmon
                   + panfish + travelcost | 0, data)
summary(model_1)

#Q2.ii: preference heterogenerity r.t. person_specific variables
data$panfish_kids <- data$panfish * data$kids
data$restroom_kids <- data$restroom * data$kids
data$ramp_boat <- data$ramp * data$boat
data$walleye_boat <- data$walleye * data$boat

#write.csv(data, file="data_forloop.csv")

model_2 <- mlogit (choice ~ ramp + restroom + walleye + salmon + panfish
                   + travelcost + panfish_kids + restroom_kids
                   + ramp_boat + walleye_boat | 0, data)
summary(model_2)

#Q2.iii: include an unobserved  site attribute
library(mlogit)
library(dplyr)
library(maxLik)
setwd("/Volumes/USB30FD/821_ps")
data <-read.csv("data_forloop.csv")
data <- data[-c(1)]
data$alt_id <-as.numeric(as.character(data$alt_id))
data$id <-as.numeric(as.character(data$id))
##small loop
delta = 1e-07 #tolerancel level for first loop to stop
a <-  c(-0.1030564, -0.1632874, 0.3442013, 0.9818902, 0.6961032)

theta_1 <- replicate(100,0)
theta_2 <- replicate(100,0)
data$theta_j = 0
data$choice <- ifelse(data$choice == "FALSE", 0, 1)
#function that produce the prediction of share S_j for all j
maxlikelihood <- c(2000000000)
b <-c(0)
##
repeat{
  data$Prob_ij_nominator <- exp(0 + a[1]*data$travelcost
+a[2]*data$panfish_kids + a[3]*data$restroom_kids+a[4]*data$ramp_boat +
a[5]*data$walleye_boat)
```

```
  data <- data %>%
    group_by(id) %>%
    mutate(Prob_ij_denom = sum(Prob_ij_nominator))%>%
    ungroup()

  data$Prob_ij <- data$Prob_ij_nominator / data$Prob_ij_denom
  data <- data %>%
    group_by(alt_id) %>%
    mutate(share_j = mean(Prob_ij))%>%
    ungroup()

  share <- data$share_j[1:100]

  real_share <- data$shares[1:100]

  for (j in 1:100) {
    theta_2[j] <- theta_1[j] + log(real_share[j]) - log(share[j])
  }

  theta_diff <- theta_2 - theta_1



  while (max(abs(theta_diff))>delta){
    theta_1 <- theta_2 #update new thetas
    data$theta_j <- theta_1[data$alt_id]
    data <- data %>%
      mutate(Prob_ij_nominator = exp(data$theta_j + a[1]*data$travelcost
+a[2]*data$panfish_kids + a[3]*data$restroom_kids+a[4]*data$ramp_boat +
a[5]*data$walleye_boat))%>%
      group_by(id) %>%
      mutate(Prob_ij_denom = sum(Prob_ij_nominator))%>%
      ungroup()
    data$Prob_ij <- data$Prob_ij_nominator / data$Prob_ij_denom
    data <- data %>%
      group_by(alt_id) %>%
      mutate(share_j = mean(Prob_ij)) %>%
      ungroup()
    share <- data$share_j[1:100]
    for (j in 1:100) {
      theta_2[j] <- theta_1[j] + log(real_share[j]) - log(share[j])
    }
    theta_diff <- theta_2 - theta_1
    #print(theta_2)
  }

  # theta_2 now is the optimised baseline utility for the specific "a"
parameter set.
  # level normalization: substract the mean
  level_norm <- function (x) {
    scale(x, scale = FALSE)
  }
```

```
  baseline_util <- level_norm(theta_2)
  ##update the new theta_j to the dataframe
  data$theta_j <- baseline_util[data$alt_id]

  #x6<- replicate(2404, baseline_util)

  ## calculate likelihood function based on the baseline_utility and
parameter set

  #changed here!!!! be careful
  data <- data %>%
    mutate(Prob_ij_nominator = exp(data$theta_j + a[1]*data$travelcost
+a[2]*data$panfish_kids + a[3]*data$restroom_kids+a[4]*data$ramp_boat +
a[5]*data$walleye_boat))%>%
    group_by(id) %>%
    mutate(Prob_ij_denom = sum(Prob_ij_nominator))%>%
    ungroup()
  data$Prob_ij <- data$Prob_ij_nominator / data$Prob_ij_denom
  #
  data$Prob_sitechoice <- log(data$Prob_ij^data$choice)
  LogL_attributes <- data$Prob_sitechoice
  llmax <- sum(LogL_attributes)
  maxlikelihood[-llmax < maxlikelihood] <- llmax

  ## update parameter set by conducting the maximisation loglikelihood
  #compute loglikelood function  x1, x2, x3, x4, x5, x6

  #############################

  loglike = function(theta, x) {
    # theta parameter vector; x
    mu1 = theta[1]
    mu2 = theta[2]
    mu3 = theta[3]
    mu4 = theta[4]
    mu5 = theta[5]
    x[, 20] <- exp( x[, 19] + mu1*x[, 8] +mu2*x[, 15] + mu3*x[, 16]+mu4*x[,
17] + mu5*x[, 18])
    x1 <- as.vector(unlist(x[,20]))
    x2 <- unname(tapply(x1,(seq_along(x1)-1) %/% 100, sum))
    x[, 21] <- cbind(rep(x2, each=100))
    x[, 22] <- x[, 20] / x[, 21]
    x[, 24]<- x[, 22]^x[, 3]
    ans = sum(log(x[, 24]))
    return(ans)
  }
  theta.start = a
  names(theta.start) = c("mu1", "mu2", "mu3", "mu4", "mu5")
  theta.mle = maxLik (loglike, start=theta.start, x=data, method = "BFGS")
  #theta.mle = optim(par=theta.start, fn=loglike, x=data, method ="BFGS")
  summary(theta.mle)
  summary(theta.mle$coef)
```

```
  # write an additional loop for the first stage to converge
  # print(theta.mle$estimate)
  print(theta.mle$maximum)
  a[theta.mle$maximum > maxlikelihood] <- theta.mle$estimate
  b[round(theta.mle$maximum, 3) == round(maxlikelihood, 3)] <- 1
  maxlikelihood[round(theta.mle$maximum, 3) > round(maxlikelihood, 3)] <-
theta.mle$maximum
  print(a)
  if( b > 0){
    break
  }
}



names(a) <- c("travelcost", "panfish_kids", "restroom_kids", "ramp_boat",
"walleye_boat")

print (maxlikelihood)
print(a)
###BLP second stage
##
BLP2 <- lm(theta_j ~ ramp + restroom + walleye + salmon + panfish + 0,
data=data)
summary(BLP2)

##############################################################################################
##############################################################################################
###Q2.iv
#clean the global environment for the new question and upload package and
settings we need
rm(list=ls())
library(mlogit)
library(dplyr)
library(maxLik)
setwd("C:/821_ps")
data <-read.csv("data_forloop.csv")
data <- data[-c(1)]
data$alt_id <-as.numeric(as.character(data$alt_id))
data$id <-as.numeric(as.character(data$id))
##small loop
delta = 1e-07 #tolerancel level for first loop to stop
theta_1 <- replicate(100,0)
theta_2 <- replicate(100,0)
data$theta_j = 0
data$choice <- ifelse(data$choice == "FALSE", 0, 1)
#function that produce the prediction of share S_j for all j
maxlikelihood <- c(2000000000)
b <-c(0)
a <-  c(-0.1230331, -0.1408312,  0.5068775,  1.3974301,  0.5928853 , 1) ##1
is the initial parameter setting for normal distribution of random
assignment
##add a random component in preferences for walleye
```

```
## and we also take the parameter set from Q2.iii for the initial guess.
The initial guess for delta_j is still zero for all j
## generate random value rnorm(2404, mean=0, sd=a[6])

repeat{
  random <- rnorm(2404, mean=0, sd=a[5])
  data$random <- rep(random, each=100)
  data$Prob_ij_nominator <- exp(0 + a[1]*data$travelcost
+a[2]*data$panfish_kids + a[3]*data$restroom_kids+a[4]*data$ramp_boat +
a[5]*data$walleye_boat + data$random*data$walleye)

  data <- data %>%
    group_by(id) %>%
    mutate(Prob_ij_denom = sum(Prob_ij_nominator))%>%
    ungroup()

  data$Prob_ij <- data$Prob_ij_nominator / data$Prob_ij_denom
  data <- data %>%
    group_by(alt_id) %>%
    mutate(share_j = mean(Prob_ij))%>%
    ungroup()

  share <- data$share_j[1:100]

  real_share <- data$shares[1:100]

  for (j in 1:100) {
    theta_2[j] <- theta_1[j] + log(real_share[j]) - log(share[j])
  }

  theta_diff <- theta_2 - theta_1



  while (max(abs(theta_diff))>delta){
    theta_1 <- theta_2 #update new thetas
    data$theta_j <- theta_1[data$alt_id]
    data <- data %>%
      mutate(Prob_ij_nominator = exp(data$theta_j + a[1]*data$travelcost
+a[2]*data$panfish_kids + a[3]*data$restroom_kids+a[4]*data$ramp_boat +
a[5]*data$walleye_boat + data$random*data$walleye))%>%
      group_by(id) %>%
      mutate(Prob_ij_denom = sum(Prob_ij_nominator))%>%
      ungroup()
    data$Prob_ij <- data$Prob_ij_nominator / data$Prob_ij_denom
    data <- data %>%
      group_by(alt_id) %>%
      mutate(share_j = mean(Prob_ij)) %>%
      ungroup()
    share <- data$share_j[1:100]
    for (j in 1:100) {
      theta_2[j] <- theta_1[j] + log(real_share[j]) - log(share[j])
    }
    theta_diff <- theta_2 - theta_1
```

```
    print(theta_2)
  }

  # theta_2 now is the optimised baseline utility for the specific "a"
parameter set.
  # level normalization: substract the mean
  level_norm <- function (x) {
    scale(x, scale = FALSE)
  }

  baseline_util <- level_norm(theta_2)
  ##update the new theta_j to the dataframe
  data$theta_j <- baseline_util[data$alt_id]

  #x6<- replicate(2404, baseline_util)

  ## calculate likelihood function based on the baseline_utility and
parameter set

  data <- data %>%
    mutate(Prob_ij_nominator = exp(data$theta_j + a[1]*data$travelcost
+a[2]*data$panfish_kids + a[3]*data$restroom_kids+a[4]*data$ramp_boat +
a[5]*data$walleye_boat + data$random*data$walleye))%>%
    group_by(id) %>%
    mutate(Prob_ij_denom = sum(Prob_ij_nominator))%>%
    ungroup()
  data$Prob_ij <- data$Prob_ij_nominator / data$Prob_ij_denom
  #
  data$Prob_sitechoice <- log(data$Prob_ij^data$choice)
  LogL_attributes <- data$Prob_sitechoice
  llmax <- sum(LogL_attributes)
  maxlikelihood[-llmax < maxlikelihood] <- llmax

  ## update parameter set by conducting the maximisation loglikelihood
  #compute loglikelood function  x1, x2, x3, x4, x5, x6

  #############################

  loglike = function(theta, x) {
    # theta parameter vector; x
    mu1 = theta[1]
    mu2 = theta[2]
    mu3 = theta[3]
    mu4 = theta[4]
    mu5 = theta[5]
    x[, 21] <- exp( x[, 19] + mu1*x[, 8] +mu2*x[, 15] + mu3*x[, 16]+mu4*x[,
17] + mu5*x[, 18] + x[,20]*x[,11])
    x1 <- as.vector(unlist(x[,21]))
    x2 <- unname(tapply(x1,(seq_along(x1)-1) %/% 100, sum))
    x[, 22] <- cbind(rep(x2, each=100))
    x[, 23] <- x[, 21] / x[, 22]
    x[, 25]<- x[, 23]^x[, 3]
    ans = sum(log(x[, 25]))
    return(ans)
```

```
  }

  theta.start = a
  names(theta.start) = c("mu1", "mu2", "mu3", "mu4", "mu5", "sigma")
  theta.mle = maxLik (loglike, start=theta.start, x=data, method = "BFGS")
  #theta.mle = optim(par=theta.start, fn=loglike, x=data, method ="BFGS")
  summary(theta.mle)
  summary(theta.mle$coef)


  # write an additional loop for the first stage to converge
  # print(theta.mle$estimate)
  print(theta.mle$maximum)
  a[theta.mle$maximum > maxlikelihood] <- theta.mle$estimate
  a[6] <- sd(data$random)
  b[round(theta.mle$maximum, 3) == round(maxlikelihood, 3)] <- 1
  maxlikelihood[theta.mle$maximum > maxlikelihood] <- theta.mle$maximum
  print(a)
  if( b > 0){
    break
  }
}



print(a)
print (llmax)
print(theta.mle$estimate)



###BLP second stage
##
BLP2 <- lm(theta_j ~ ramp + restroom + walleye + salmon + panfish + 0,
data=data)
summary(BLP2)

###################Q3
library(mlogit)
library(dplyr)
library(maxLik)
setwd("/Volumes/USB30FD/821_ps")
#####data upload and prepare
data <- read.csv("long_data.csv")
data <- data[-c(1)]
names(data)[names(data) == "idcase"] <- "id"
data <- mlogit.data(data, choice = "choice",
                    shape = "long", alt.levels = (c(1:100)), id ="id")
data$shares <- data$shares*100
names(data)[names(data) == "shares"] <- "shares_100"
#Q3.1i:try the simple model first
model_1 <- mlogit (choice ~ ramp + restroom + walleye + salmon
                   + panfish + travelcost + shares_100| 0, data)
```

```
summary(model_1)


#Q3.1ii: preference heterogenerity r.t. person_specific variables
data$panfish_kids <- data$panfish * data$kids
data$restroom_kids <- data$restroom * data$kids
data$ramp_boat <- data$ramp * data$boat
data$walleye_boat <- data$walleye * data$boat

model_2 <- mlogit (choice ~ ramp + restroom + walleye + salmon + panfish
                   + travelcost + panfish_kids + restroom_kids
                   + ramp_boat + walleye_boat + shares_100 | 0, data)
summary(model_2)



#Q3.1iii: BLP
data$alt_id <-as.numeric(as.character(data$alt_id))
data$id <-as.numeric(as.character(data$id))
##small loop
delta = 1e-07 #tolerancel level for first loop to stop
a <-  c(0, 0, 0, 0, 0)

theta_1 <- replicate(100,0)
theta_2 <- replicate(100,0)
data$theta_j = 0
data$choice <- ifelse(data$choice == "FALSE", 0, 1)
#function that produce the prediction of share S_j for all j
maxlikelihood <- c(2000000000)
b <-c(0)
##
repeat{
  data$Prob_ij_nominator <- exp(0 + a[1]*data$travelcost
+a[2]*data$panfish_kids + a[3]*data$restroom_kids+a[4]*data$ramp_boat +
a[5]*data$walleye_boat)

  data <- data %>%
    group_by(id) %>%
    mutate(Prob_ij_denom = sum(Prob_ij_nominator))%>%
    ungroup()

  data$Prob_ij <- data$Prob_ij_nominator / data$Prob_ij_denom
  data <- data %>%
    group_by(alt_id) %>%
    mutate(share_j = mean(Prob_ij)*100)%>%
    ungroup()

  share <- data$share_j[1:100]

  real_share <- data$shares_100[1:100]

  for (j in 1:100) {
    theta_2[j] <- theta_1[j] + log(real_share[j]) - log(share[j])
  }
```

```
  theta_diff <- theta_2 - theta_1



  while (max(abs(theta_diff))>delta){
    theta_1 <- theta_2 #update new thetas
    data$theta_j <- theta_1[data$alt_id]
    data <- data %>%
      mutate(Prob_ij_nominator = exp(data$theta_j + a[1]*data$travelcost
+a[2]*data$panfish_kids + a[3]*data$restroom_kids+a[4]*data$ramp_boat +
a[5]*data$walleye_boat))%>%
      group_by(id) %>%
      mutate(Prob_ij_denom = sum(Prob_ij_nominator))%>%
      ungroup()
    data$Prob_ij <- data$Prob_ij_nominator / data$Prob_ij_denom
    data <- data %>%
      group_by(alt_id) %>%
      mutate(share_j = mean(Prob_ij)*100) %>%
      ungroup()
    share <- data$share_j[1:100]
    for (j in 1:100) {
      theta_2[j] <- theta_1[j] + log(real_share[j]) - log(share[j])
    }
    theta_diff <- theta_2 - theta_1
    #print(theta_2)
  }

  # theta_2 now is the optimised baseline utility for the specific "a"
parameter set.
  # level normalization: substract the mean
  level_norm <- function (x) {
    scale(x, scale = FALSE)
  }

  baseline_util <- level_norm(theta_2)
  ##update the new theta_j to the dataframe
  data$theta_j <- baseline_util[data$alt_id]

  #x6<- replicate(2404, baseline_util)

  ## calculate likelihood function based on the baseline_utility and
parameter set

  #changed here!!!! be careful
  data <- data %>%
    mutate(Prob_ij_nominator = exp(data$theta_j + a[1]*data$travelcost
+a[2]*data$panfish_kids + a[3]*data$restroom_kids+a[4]*data$ramp_boat +
a[5]*data$walleye_boat))%>%
    group_by(id) %>%
    mutate(Prob_ij_denom = sum(Prob_ij_nominator))%>%
    ungroup()
  data$Prob_ij <- data$Prob_ij_nominator / data$Prob_ij_denom
  #
```

```
  data$Prob_sitechoice <- log(data$Prob_ij^data$choice)
  LogL_attributes <- data$Prob_sitechoice
  llmax <- sum(LogL_attributes)
  maxlikelihood[-llmax < maxlikelihood] <- llmax

  ## update parameter set by conducting the maximisation loglikelihood
  #compute loglikelood function  x1, x2, x3, x4, x5, x6

  #############################

  loglike = function(theta, x) {
    # theta parameter vector; x
    mu1 = theta[1]
    mu2 = theta[2]
    mu3 = theta[3]
    mu4 = theta[4]
    mu5 = theta[5]
    x[, 20] <- exp( x[, 19] + mu1*x[, 8] +mu2*x[, 15] + mu3*x[, 16]+mu4*x[,
17] + mu5*x[, 18])
    x1 <- as.vector(unlist(x[,20]))
    x2 <- unname(tapply(x1,(seq_along(x1)-1) %/% 100, sum))
    x[, 21] <- cbind(rep(x2, each=100))
    x[, 22] <- x[, 20] / x[, 21]
    x[, 24]<- x[, 22]^x[, 3]
    ans = sum(log(x[, 24]))
    return(ans)
  }
  theta.start = a
  names(theta.start) = c("mu1", "mu2", "mu3", "mu4", "mu5")
  theta.mle = maxLik (loglike, start=theta.start, x=data, method = "BFGS")
  #theta.mle = optim(par=theta.start, fn=loglike, x=data, method ="BFGS")
  summary(theta.mle)
  summary(theta.mle$coef)


  # write an additional loop for the first stage to converge
  # print(theta.mle$estimate)
  print(theta.mle$maximum)
  a[theta.mle$maximum > maxlikelihood] <- theta.mle$estimate
  b[round(theta.mle$maximum, 3) == round(maxlikelihood, 3)] <- 1
  maxlikelihood[round(theta.mle$maximum, 3) > round(maxlikelihood, 3)] <-
theta.mle$maximum
  print(a)
  if( b > 0){
    break
  }
}


names(a) <- c("travelcost", "panfish_kids", "restroom_kids", "ramp_boat",
"walleye_boat")
print(a)
print (maxlikelihood)
print(theta.mle$estimate)
```

```
##> print(a)
##[1] -0.1230306 -0.  1408963  0.5048161  1.3881009  0.5524893
##> print (maxlikelihood)
##[1] -5136.971
##> print(theta.mle$estimate)
##mu1        mu2        mu3        mu4        mu5
##-0.1230306 -0.1408963  0.5048161  1.3881009  0.5524893

###BLP second stage
##
BLP2 <- lm(theta_j ~ ramp + restroom + walleye + salmon + panfish +
shares_100 + 0, data=data)
summary(BLP2)
###########################################################################################

#Q3.2
library(quantreg)
library(gmm)
###arange data frame
data_iv <- data[1:100 ,]
data_iv <- subset(data_iv, select=c("ramp", "restroom",
"walleye","salmon","panfish","shares_100","theta_j"))
############generate instruments
###median regression
rqfit <- rq(theta_j ~ ramp + restroom + walleye + salmon + panfish +
shares_100, data=data_iv)
coef <-rqfit$coefficients
###calculate shares as one of instruments
data$Prob_ij_nominator <- exp(coef[1] + coef[2]*data$ramp +
coef[3]*data$restroom + coef[4]*data$walleye + coef[5]*data$salmon +
coef[6]*data$panfish + a[1]*data$travelcost +a[2]*data$panfish_kids +
a[3]*data$restroom_kids+a[4]*data$ramp_boat + a[5]*data$walleye_boat)

data <- data %>%
  group_by(id) %>%
  mutate(Prob_ij_denom = sum(Prob_ij_nominator))%>%
  ungroup()

data$Prob_ij <- data$Prob_ij_nominator / data$Prob_ij_denom
data <- data %>%
  group_by(alt_id) %>%
  mutate(share_j = mean(Prob_ij))%>%
  ungroup()

data_iv <- data[1:100 ,]
data_iv <- subset(data_iv, select=c("ramp", "restroom",
"walleye","salmon","panfish","shares_100","theta_j","share_j"))
data_iv$share_j <- data_iv$share_j*100
####quantile IV GMM
##generate condition function
g1 <- function (tet, x){
  #tet <- parameter set
  #x <- data_iv dataframe
```

```
  #Sn <- 0.25, same as the setting of orginal paper
  # intercept from median regression <- -0.1190958
  m1 <- (pnorm((x[,7] + 0.1190958 - tet[1]*x[,1] - tet[2]*x[,2] -
tet[3]*x[,3] - tet[4]*x[,4] - tet[5]*x[,5] - tet[6]*x[,8])/0.25) - 0.5 )
  return(m1)
}

####other method: "inverse" quantile estimation
install.packages("remotes")
remotes::install_github("yuchang0321/IVQR")
library(IVQR)

fit <- ivqr(theta_j~ shares_100 | share_j | ramp + restroom + walleye +
salmon + panfish, 0.5, grid= seq(-4.5 , 0 , 0.05625), data = data_iv)
fit
####Other Method: 2sls
#OLS
ols<- lm(theta_j ~ ramp + restroom + walleye + salmon + panfish +
shares_100 + 0, data=data_iv)
coef <-ols$coefficients
#predict shares of visiting as instrument, based on exogenous things only
data$Prob_ij_nominator <- exp( coef[1]*data$ramp + coef[2]*data$restroom +
coef[3]*data$walleye + coef[4]*data$salmon + coef[5]*data$panfish +
a[1]*data$travelcost +a[2]*data$panfish_kids +
a[3]*data$restroom_kids+a[4]*data$ramp_boat + a[5]*data$walleye_boat)

data <- data %>%
  group_by(id) %>%
  mutate(Prob_ij_denom = sum(Prob_ij_nominator))%>%
  ungroup()

data$Prob_ij <- data$Prob_ij_nominator / data$Prob_ij_denom
data <- data %>%
  group_by(alt_id) %>%
  mutate(share_j = mean(Prob_ij))%>%
  ungroup()

data_iv <- data[1:100 ,]
data_iv <- subset(data_iv, select=c("ramp", "restroom",
"walleye","salmon","panfish","shares_100","theta_j","share_j"))
data_iv$share_j <- data_iv$share_j*100

#2SLS
library(ivpack)
twosls <- ivreg(data_iv$theta_j~ data_iv$shares_100 + data_iv$ramp +
data_iv$restroom +data_iv$walleye +data_iv$salmon + data_iv$panfish |
data_iv$share_j + data_iv$ramp + data_iv$restroom +data_iv$walleye
+data_iv$salmon + data_iv$panfish   )
beta <- c(-3.7598, -0.3720, -0.4888, 8.1968, 20.3990, 1.2842) ##from Q3_2
names(beta) <- c("shares_100" , "ramp", "restroom", "walleye", "salmon",
"panfish")
data_iv$unobserved <- data_iv$theta_j - beta[1]*data_iv$shares_100 -
beta[2]*data_iv$ramp - beta[3]*data_iv$restroom -beta[4]*data_iv$walleye -
beta[5]*data_iv$salmon - beta[6]*data_iv$panfish
```

```
####################################################
##data preparation for Q4
unobserved <- data_iv$unobserved

data <- read.csv("long_data.csv")
data <- data[-c(1)]
names(data)[names(data) == "idcase"] <- "id"
data <- mlogit.data(data, choice = "choice",
                    shape = "long", alt.levels = (c(1:100)), id ="id")
data$shares <- data$shares*100
names(data)[names(data) == "shares"] <- "shares_100"
data$panfish_kids <- data$panfish * data$kids
data$restroom_kids <- data$restroom * data$kids
data$ramp_boat <- data$ramp * data$boat
data$walleye_boat <- data$walleye * data$boat
data$alt_id <-as.numeric(as.character(data$alt_id))
data$id <-as.numeric(as.character(data$id))
data$unobserved <- cbind(rep( unobserved, 2404))
write.csv(data, file = "data_for_welfare_analysis.csv")

###### Welfare Analysis
##data
library(dplyr)
setwd("C:/821_ps")
data <- read.csv("data_for_welfare_analysis.csv")
data <- data[-c(1)]
data$id <-as.numeric(as.character(data$id))

##generate utility calculation function
beta <- c(-3.7598, -0.3720, -0.4888, 8.1968, 20.3990, 1.2842) ##from Q3_2
names(beta) <- c("shares_100" , "ramp", "restroom", "walleye", "salmon",
"panfish")
alpha <- c(-0.1230306, -0.1408963,  0.5048161,  1.3881009,  0.5524893 )
names(alpha) <- c("travelcost", "panfish_kids", "restroom_kids",
"ramp_boat", "walleye_boat")
U <- function(beta, alpha, x) {
  ans <- beta[1]*x[,14] + beta[2]*x[,9] + beta[3]*x[,10]+ beta[4]*x[,11] +
beta[5]*x[,12] + beta[6]*x[,13] + alpha[1]*x[,8] + alpha[2]*x[,15] +
alpha[3]*x[,16] + alpha[4]*x[,17] + alpha[5]*x[,18]+ x[,19]
  return(ans)

}
#####################i Partial equilibrium
data$old_utility <- U(beta, alpha, data)
write.csv(data, file="data_welfare.csv")
### scenario A
data$walleye <- data$walleye*1.3
data$walleye_boat <-data$walleye*data$boat
data$new_utility <- U(beta, alpha, data)

data$old_utility <- exp(data$old_utility)
data$new_utility <- exp(data$new_utility)

new_utility <- data$new_utility
```

```
sum_new <- unname(tapply(new_utility,(seq_along(new_utility)-1) %/% 100,
sum))
data$sum_new <- cbind(rep(sum_new, each =100))
data$sum_new <- log(data$sum_new)

old_utility <- data$old_utility
sum_old <- unname(tapply(old_utility,(seq_along(old_utility)-1) %/% 100,
sum))
data$sum_old <- cbind(rep(sum_old, each =100))
data$sum_old <- log(data$sum_old)

data$CV <- (1/0.123131)*(data$sum_new-data$sum_old)
mean(data$CV)   ###  5.217842


### scenario B
data <- read.csv("data_welfare.csv")
data <- data[-c(1)]
data$change <- ifelse(data$shares_100 >1.5, 1.3, 1)
data$walleye <- data$walleye*data$change
data$walleye_boat <-data$walleye*data$boat
data$new_utility <- U(beta, alpha, data)

data$old_utility <- exp(data$old_utility)
data$new_utility <- exp(data$new_utility)

new_utility <- data$new_utility
sum_new <- unname(tapply(new_utility,(seq_along(new_utility)-1) %/% 100,
sum))
data$sum_new <- cbind(rep(sum_new, each =100))
data$sum_new <- log(data$sum_new)

old_utility <- data$old_utility
sum_old <- unname(tapply(old_utility,(seq_along(old_utility)-1) %/% 100,
sum))
data$sum_old <- cbind(rep(sum_old, each =100))
data$sum_old <- log(data$sum_old)
data$CV <- (1/0.123131)*(data$sum_new-data$sum_old)

mean_affected <- data$CV[which(data$change == 1.3 & data$choice =="TRUE") ]
%>% mean() ###5.09357
mean_unaffected <- mean(data$CV[which(data$change == 1.0 & data$choice
=="TRUE") ]) %>% mean() ###2.993604


### scenario C
data <- read.csv("data_welfare.csv")
data <- data[-c(1)]
data <- data[which(data$shares_100 <= 1.5),] ##now we have 82 sites

data$new_utility <- U(beta, alpha, data)

data$old_utility <- exp(data$old_utility)
data$new_utility <- exp(data$new_utility)
```

```
new_utility <- data$new_utility
sum_new <- unname(tapply(new_utility,(seq_along(new_utility)-1) %/% 82,
sum))
data$sum_new <- cbind(rep(sum_new, each =82))
data$sum_new <- log(data$sum_new)

old_utility <- data$old_utility
sum_old <- unname(tapply(old_utility,(seq_along(old_utility)-1) %/% 82,
sum))
data$sum_old <- cbind(rep(sum_old, each =82))
data$sum_old <- log(data$sum_old)
data$CV <- (1/0.123131)*(data$sum_new-data$sum_old)

mean_unaffected_by_removal <- data$CV[which(data$choice =="TRUE") ] %>%
mean() ###9.157485e-17

### scenario D
data <- read.csv("data_welfare.csv")
data <- data[-c(1)]
data$change <- ifelse(data$shares_100 >1.5, 10, 0)
data$travelcost <- data$travelcost+data$change

data$new_utility <- U(beta, alpha, data)

data$old_utility <- exp(data$old_utility)
data$new_utility <- exp(data$new_utility)

new_utility <- data$new_utility
sum_new <- unname(tapply(new_utility,(seq_along(new_utility)-1) %/% 100,
sum))
data$sum_new <- cbind(rep(sum_new, each =100))
data$sum_new <- log(data$sum_new)

old_utility <- data$old_utility
sum_old <- unname(tapply(old_utility,(seq_along(old_utility)-1) %/% 100,
sum))
data$sum_old <- cbind(rep(sum_old, each =100))
data$sum_old <- log(data$sum_old)
data$CV <- (1/0.123131)*(data$sum_new-data$sum_old)

mean_affected <- data$CV[which(data$change == 10 & data$choice =="TRUE") ]
%>% mean() ###-4.563843
mean_unaffected <- mean(data$CV[which(data$change == 0 & data$choice
=="TRUE") ]) %>% mean() ###-2.341332


######################ii general equilibrium <- resorting

### scenario A
data <- read.csv("data_welfare.csv")
data <- data[-c(1)]
data$walleye <- data$walleye*1.3
data$walleye_boat <-data$walleye*data$boat
```

```
#stimulate new choice set
data$nominator <- exp(U(beta, alpha, data))
nominator <- data$nominator
denominator <- unname(tapply(nominator,(seq_along(nominator)-1) %/% 100,
sum))
data$denomitor <- cbind(rep(denominator, each =100))
data$pij <- data$nominator/data$denomitor
data<-arrange(data, data$alt_id)
pij <- data$pij
pij_mean <- unname(tapply(pij,(seq_along(pij)-1) %/% 2404, sum))
data$shares_new <- cbind(rep(pij_mean, each =2404))
data<-arrange(data, data$id)
##calculate new utility
U_GE <- function(beta, alpha, x) {
  ans <- beta[1]*x[,24] + beta[2]*x[,9] + beta[3]*x[,10]+ beta[4]*x[,11] +
beta[5]*x[,12] + beta[6]*x[,13] + alpha[1]*x[,8] + alpha[2]*x[,15] +
alpha[3]*x[,16] + alpha[4]*x[,17] + alpha[5]*x[,18]+ x[,19]
  return(ans)
}
data$new_utility <- U_GE(beta, alpha, data)

data$old_utility <- exp(data$old_utility)
data$new_utility <- exp(data$new_utility)

new_utility <- data$new_utility
sum_new <- unname(tapply(new_utility,(seq_along(new_utility)-1) %/% 100,
sum))
data$sum_new <- cbind(rep(sum_new, each =100))
data$sum_new <- log(data$sum_new)

old_utility <- data$old_utility
sum_old <- unname(tapply(old_utility,(seq_along(old_utility)-1) %/% 100,
sum))
data$sum_old <- cbind(rep(sum_old, each =100))
data$sum_old <- log(data$sum_old)

data$CV <- (1/0.123131)*(data$sum_new-data$sum_old)
mean(data$CV)   ###  -147.337

### scenario B
data <- read.csv("data_welfare.csv")
data <- data[-c(1)]
data$change <- ifelse(data$shares_100 >1.5, 1.3, 1)
data$walleye <- data$walleye*data$change
data$walleye_boat <-data$walleye*data$boat
#stimulate new choice set
data$nominator <- exp(U(beta, alpha, data))
nominator <- data$nominator
denominator <- unname(tapply(nominator,(seq_along(nominator)-1) %/% 100,
sum))
data$denomitor <- cbind(rep(denominator, each =100))
data$pij <- data$nominator/data$denomitor
data<-arrange(data, data$alt_id)
pij <- data$pij
```

```
pij_mean <- unname(tapply(pij,(seq_along(pij)-1) %/% 2404, sum))
data$shares_new <- cbind(rep(pij_mean, each =2404))
data<-arrange(data, data$id)
##calculate new utility
U_GE <- function(beta, alpha, x) {
  ans <- beta[1]*x[,25] + beta[2]*x[,9] + beta[3]*x[,10]+ beta[4]*x[,11] +
beta[5]*x[,12] + beta[6]*x[,13] + alpha[1]*x[,8] + alpha[2]*x[,15] +
alpha[3]*x[,16] + alpha[4]*x[,17] + alpha[5]*x[,18]+ x[,19]
  return(ans)
}
data$new_utility <- U_GE(beta, alpha, data)

data$old_utility <- exp(data$old_utility)
data$new_utility <- exp(data$new_utility)

new_utility <- data$new_utility
sum_new <- unname(tapply(new_utility,(seq_along(new_utility)-1) %/% 100,
sum))
data$sum_new <- cbind(rep(sum_new, each =100))
data$sum_new <- log(data$sum_new)

old_utility <- data$old_utility
sum_old <- unname(tapply(old_utility,(seq_along(old_utility)-1) %/% 100,
sum))
data$sum_old <- cbind(rep(sum_old, each =100))
data$sum_old <- log(data$sum_old)
data$CV <- (1/0.123131)*(data$sum_new-data$sum_old)
## update actual choice
data <- arrange(data, id)
new_utility <- c(data$new_utility)
data$new_utility <- new_utility
data<- arrange(data, id, desc(data$new_utility))
data$new_choice = "FALSE"
for (i in 0:2403 ){
  data$new_choice[(100*i+1)] <- "TRUE"
}

mean_affected <- data$CV[which(data$change == 1.3 & data$new_choice
=="TRUE") ] %>% mean() ###NaN, suggesting no person change to the place
when walleye increased in the crowded place.
mean_unaffected <- mean(data$CV[which(data$change == 1.0 & data$new_choice
=="TRUE") ]) %>% mean() ###-144.6079


###Scenerio C
data <- read.csv("data_welfare.csv")
data <- data[-c(1)]
data <- data[which(data$shares_100 <= 1.5),] ##now we have 82 sites
#stimulate new choice set
data$nominator <- exp(U(beta, alpha, data))
nominator <- data$nominator
denominator <- unname(tapply(nominator,(seq_along(nominator)-1) %/% 82,
sum))
data$denomitor <- cbind(rep(denominator, each =82))
```

```
data$pij <- data$nominator/data$denomitor
data<-arrange(data, data$alt_id)
pij <- data$pij
pij_mean <- unname(tapply(pij,(seq_along(pij)-1) %/% 2404, sum))
data$shares_new <- cbind(rep(pij_mean, each =2404))
data<-arrange(data, data$id)
##calculate new utility
U_GE <- function(beta, alpha, x) {
  ans <- beta[1]*x[,24] + beta[2]*x[,9] + beta[3]*x[,10]+ beta[4]*x[,11] +
beta[5]*x[,12] + beta[6]*x[,13] + alpha[1]*x[,8] + alpha[2]*x[,15] +
alpha[3]*x[,16] + alpha[4]*x[,17] + alpha[5]*x[,18]+ x[,19]
  return(ans)
}
data$new_utility <- U_GE(beta, alpha, data)

data$old_utility <- exp(data$old_utility)
data$new_utility <- exp(data$new_utility)

new_utility <- data$new_utility
sum_new <- unname(tapply(new_utility,(seq_along(new_utility)-1) %/% 82,
sum))
data$sum_new <- cbind(rep(sum_new, each =82))
data$sum_new <- log(data$sum_new)

old_utility <- data$old_utility
sum_old <- unname(tapply(old_utility,(seq_along(old_utility)-1) %/% 82,
sum))
data$sum_old <- cbind(rep(sum_old, each =82))
data$sum_old <- log(data$sum_old)
data$CV <- (1/0.123131)*(data$sum_new-data$sum_old)

mean_unaffected_by_removal <- data$CV[which(data$choice =="TRUE") ] %>%
mean() ###-350.9764

### scenario D
data <- read.csv("data_welfare.csv")
data <- data[-c(1)]
data$change <- ifelse(data$shares_100 >1.5, 10, 0)
data$travelcost <- data$travelcost+data$change
#stimulate new choice set
data$nominator <- exp(U(beta, alpha, data))
nominator <- data$nominator
denominator <- unname(tapply(nominator,(seq_along(nominator)-1) %/% 100,
sum))
data$denomitor <- cbind(rep(denominator, each =100))
data$pij <- data$nominator/data$denomitor
data<-arrange(data, data$alt_id)
pij <- data$pij
pij_mean <- unname(tapply(pij,(seq_along(pij)-1) %/% 2404, sum))
data$shares_new <- cbind(rep(pij_mean, each =2404))
data<-arrange(data, data$id)
##calculate new utility
U_GE <- function(beta, alpha, x) {
```

```
  ans <- beta[1]*x[,25] + beta[2]*x[,9] + beta[3]*x[,10]+ beta[4]*x[,11] +
beta[5]*x[,12] + beta[6]*x[,13] + alpha[1]*x[,8] + alpha[2]*x[,15] +
alpha[3]*x[,16] + alpha[4]*x[,17] + alpha[5]*x[,18]+ x[,19]
  return(ans)
}
data$new_utility <- U_GE(beta, alpha, data)

data$old_utility <- exp(data$old_utility)
data$new_utility <- exp(data$new_utility)

new_utility <- data$new_utility
sum_new <- unname(tapply(new_utility,(seq_along(new_utility)-1) %/% 100,
sum))
data$sum_new <- cbind(rep(sum_new, each =100))
data$sum_new <- log(data$sum_new)

old_utility <- data$old_utility
sum_old <- unname(tapply(old_utility,(seq_along(old_utility)-1) %/% 100,
sum))
data$sum_old <- cbind(rep(sum_old, each =100))
data$sum_old <- log(data$sum_old)
data$CV <- (1/0.123131)*(data$sum_new-data$sum_old)
## update actual choice
data <- arrange(data, id)
new_utility <- c(data$new_utility)
data$new_utility <- new_utility
data<- arrange(data, id, desc(data$new_utility))
data$new_choice = "FALSE"
for (i in 0:2403 ){
  data$new_choice[(100*i+1)] <- "TRUE"
}

mean_affected <- data$CV[which(data$change == 10 & data$new_choice
=="TRUE") ] %>% mean() ###NaN
mean_unaffected <- mean(data$CV[which(data$change == 0 & data$new_choice
=="TRUE") ]) %>% mean() ###-345.9996
```