# This Notebook:

"A Maximum Likelihood Bunching Estimator of the Elasticity of Taxable Income" written by Thomas Aronsson, Katharina Jenderny and Gauthier Lanot, accepted for publication in the Journal of Applied Econometrics, MS-13011.

This notebook applies the Chetty approach to the estimation of the ETI using Swedish evidence. (Month 28th since revision request...)

This file deals with "ETI_hist_CBEFVI_bin100_2019.dta".

We have first a bunch of declaration (functions etc...), then a presentation of the data, and then some calculations. The calculations are slightly more extensive than those presented in the text of the paper. This may convince the reader that we have carried out due diligence...

In [1]:
```
#Parallelism().set(nproc=6)
#print(Parallelism())
```

In [2]:
```
import numpy  as np
import pandas as pd
import os
```

In [3]:
```
tt=os.getcwd()
show(tt)
```

In [4]:
```
#hessian, for var cov+ checking optimum...
# hh0b=matrix(RDF,hes(lik0,q00b))
# hh0b.eigenvalues()
# hh0ib = hh0b.inverse()
# sqrtN =sqrt(sum(n[1] for n in nobs))
# results log-likelihood estimates...
# res_lik_print0(q00b,hh0ib,sqrtN,sum(n[1] for n in nobs),"Likelihood Estimates,
```

# Tax Info

In [5]:
```
# this is for UMea...what is the swedish average...?
taxinfo = [ [2001,0.677,0.477,252000],
          [2002,0.677,0.477,273800],
          [2003,0.677,0.477,284300],
          [2004,0.677,0.477,291800],
          [2005,0.669,0.469,298600],
          [2006,0.669,0.469,306000],
          [2007,0.669,0.469,316700],
          [2008,0.669,0.469,328800],
          [2009,0.669,0.469,367600],
          [2010,0.669,0.469,372100],
          [2011,0.669,0.469,383000],
          [2012,0.669,0.469,401100],
          [2013,0.669,0.469,413200],
          [2014,0.664,0.464,420800],
          [2015,0.664,0.464,430200],
          [2016,0.6635,0.4635,430200],
          [2017,0.6585,0.46585,438900],
          [2018,0.6585,0.46585,455300],
          [2019,0.6585,0.46585,490700]
        ]
```

In [ ]:
```
```

In [6]:
```
# so how do we select a symmetric range around the kink...?
# we check the distance from the kink to the min or the max,
# and select only if the difference from the kink
# is less that the minimum between those 2 differences... simple...
year = 2019
Info_in_year = flatten([ nn for nn in taxinfo if nn[0]==year])
Kink = Info_in_year[3]/1000.
lnk=ln(Kink).n()
```

In [7]:
```
os.chdir(tt)
tt2=os.getcwd()
show(tt2)
```

# Data:

Here comes the data.

In [8]:
```
import numpy  as np
import pandas as pd
import os
```

In [9]:
```
#newd="D://Umeå universitet//ETI - General//JAE Revisions//scb data//hist_data"
#newd = "/Users/gauthierlanot/Library/CloudStorage/OneDrive-SharedLibraries-Umeå
#newd = "//Users/gauthierlanot/Downloads/"
#ETI - General\\JAE Revisions\\scb data"
newd = "D:\ETIprojectfiles\JAE revisions"
show(newd)
```

```
<>:5: DeprecationWarning: invalid escape sequence \E
<>:5: DeprecationWarning: invalid escape sequence \E
<>:5: DeprecationWarning: invalid escape sequence \E
<ipython-input-9-69121d137286>:5: DeprecationWarning: invalid escape sequence \E
  newd = "D:\ETIprojectfiles\JAE revisions"
```

In [10]:
```
# loading data from stata file
os.chdir(newd)
filename = "ETI_hist_CBEFVI_bin100_2019.dta"
```

In [ ]:
```
```

In [11]:
```
dd0 = pd.read_stata(filename)
```

In [12]:
```
dd = dd0.to_numpy()
```

In [13]:
```
dd = [(nn[0],nn[1],nn[2],nn[3]/1000.,nn[4],nn[5],nn[6]/1000.) for nn in dd]
```

In [14]:
```
# ...and here we find that the data is not centered on the kink!
# typical last minute job! no attention to details!
#df0
```

In [15]:
```
dd[0:10]
```

Out[15]:
```
[(1.0, 1208.0, 2019.0, 422.3, 422300.0, 0.0, 490.7),
 (2.0, 1187.0, 2019.0, 422.4, 422400.0, 1.0, 490.7),
 (3.0, 1206.0, 2019.0, 422.5, 422500.0, 1.0, 490.7),
 (4.0, 1149.0, 2019.0, 422.6, 422600.0, 1.0, 490.7),
 (5.0, 1098.0, 2019.0, 422.7, 422700.0, 1.0, 490.7),
 (6.0, 1188.0, 2019.0, 422.8, 422800.0, 1.0, 490.7),
 (7.0, 1144.0, 2019.0, 422.9, 422900.0, 1.0, 490.7),
 (8.0, 1111.0, 2019.0, 423.0, 423000.0, 1.0, 490.7),
 (9.0, 1134.0, 2019.0, 423.1, 423100.0, 1.0, 490.7),
 (10.0, 1174.0, 2019.0, 423.2, 423200.0, 1.0, 490.7)]
```

In [16]:
```
# so how do we select a symmetric range around the kink...?
# we check the distance from the kink to the min or the max,
# and select only if the difference from the kink
# is less that the minimum between those 2 differences... simple...
year = 2019
Info_in_year = flatten([ nn for nn in taxinfo if nn[0]==year])
Kink = Info_in_year[3]/1000.
lnk=ln(Kink).n()
```

In [17]:
```
Info_in_year
```

Out[17]:
```
[2019, 0.658500000000000, 0.465850000000000, 490700]
```

In [18]:
```
Kink
```

Out[18]:
```
490.700000000000
```

In [19]:
```
dd[0]
```

Out[19]:
```
(1.0, 1208.0, 2019.0, 422.3, 422300.0, 0.0, 490.7)
```

In [20]:
```
#distmin2k = df0.kink[[0]].array[0]-df0.wh[[0]].array[0]
#distk2max = df0.wh[[len(df0)-1]].array[0]-df0.kink[[0]].array[0]
#topdistfromk = min(distmin2k,distk2max)+0.001
#distmin2k,distk2max,topdistfromk
distmin2k = dd[0][6]-dd[0][3]
distk2max = dd[-1][3]-dd[-1][6]
topdistfromk = min(distmin2k,distk2max)+0.001
distmin2k,distk2max,topdistfromk
```

Out[20]:
```
(68.39999999999998, 79.59999999999997, 68.40099999999998)
```

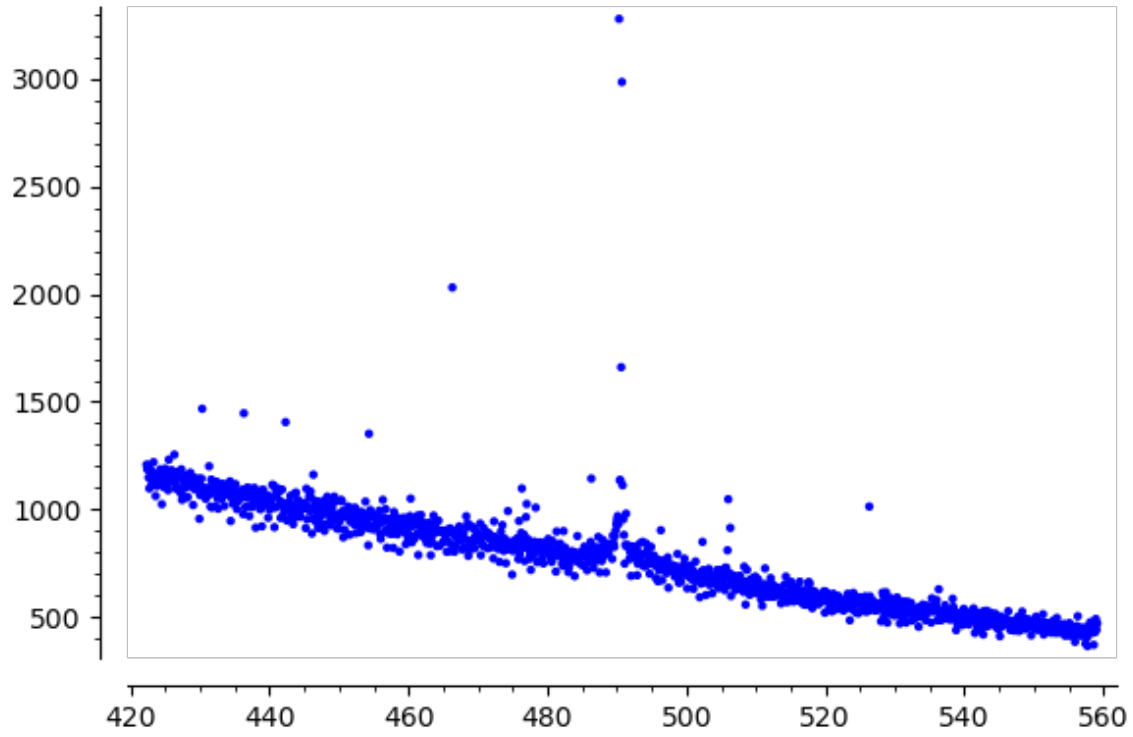
In [21]:
```
df = [nn for nn in dd if abs(nn[3]-Kink)<=topdistfromk ]
df_yy_cnt = [ (dd[3],dd[1]) for dd in df]
```

In [22]:
```
df_yy_cnt[0:10]
```

Out[22]:
```
[(422.3, 1208.0),
 (422.4, 1187.0),
 (422.5, 1206.0),
 (422.6, 1149.0),
 (422.7, 1098.0),
 (422.8, 1188.0),
 (422.9, 1144.0),
 (423.0, 1111.0),
 (423.1, 1134.0),
 (423.2, 1174.0)]
```

In [23]:
```
list_plot(df_yy_cnt)
```

Out[23]:




In [24]:
```
# transform data into array...
#df1 = df0.to_numpy()
#and select to make range centered on kink
#df = [nn for nn in df1 if abs(nn[3]-Kink)<=topdistfromk ]
df = [nn for nn in dd if abs(nn[3]-Kink)<=topdistfromk ]
```

In [25]:
```
df[0:10],lnk,Kink
```

Out[25]:
```
([(1.0, 1208.0, 2019.0, 422.3, 422300.0, 0.0, 490.7),
  (2.0, 1187.0, 2019.0, 422.4, 422400.0, 1.0, 490.7),
  (3.0, 1206.0, 2019.0, 422.5, 422500.0, 1.0, 490.7),
  (4.0, 1149.0, 2019.0, 422.6, 422600.0, 1.0, 490.7),
  (5.0, 1098.0, 2019.0, 422.7, 422700.0, 1.0, 490.7),
  (6.0, 1188.0, 2019.0, 422.8, 422800.0, 1.0, 490.7),
  (7.0, 1144.0, 2019.0, 422.9, 422900.0, 1.0, 490.7),
  (8.0, 1111.0, 2019.0, 423.0, 423000.0, 1.0, 490.7),
  (9.0, 1134.0, 2019.0, 423.1, 423100.0, 1.0, 490.7),
  (10.0, 1174.0, 2019.0, 423.2, 423200.0, 1.0, 490.7)],
 6.19583294309586,
 490.700000000000)
```

In [26]:
```
Nobs = int(sum([nn[1] for nn in df]))
show(Nobs)
dens = [(nn[3],nn[1]/Nobs) for nn in df]
dens = np.array(dens)
show(len(dens))

#densa = [(nn[3],nn[1]/Nobs/0.1) for nn in df if not(((ln(nn[3])+0.1/252.)>=lnk)

densa = [(nn[3],nn[1]/Nobs/0.1) for nn in df if nn[3]<(Kink-0.0999)]+ \
        [(nn[3],nn[1]/Nobs) for nn in df if (nn[3]>=(Kink-0.0999)) and (nn[3])<=
        [(nn[3],nn[1]/Nobs/0.1) for nn in df if nn[3]>Kink]

densb = [(nn[3],nn[1]/Nobs/0.1) for nn in df]
```

In [27]:
```
sum([nn[1] for nn in densa if nn[0]<(Kink-0.0999)])*0.1+\
sum([nn[1] for nn in densa if nn[0]>(Kink)])*0.1+\
sum([nn[1] for nn in densa if (nn[0]>=(Kink-0.0999)) and (nn[0])<=Kink])
```

Out[27]:
```
1.0
```

In [28]:
```
[nn for nn in densa if (nn[0]>=(Kink-0.1)) and (nn[0])<=Kink]
```

Out[28]:
```
[(490.6, 0.015737333388917734), (490.7, 0.0028281721954195927)]
```

In [29]:
```
type(Nobs), sum([nn[1] for nn in densb]), sum([nn[1] for nn in densa])
```

Out[29]:
```
(<class 'int'>, 10.000000000000005, 9.974546450241231)
```

In [30]:
```
[(nn[3],nn[1]/Nobs) for nn in df if (nn[3]>(Kink-0.099)) and (nn[3]-Kink)<=0]
```

Out[30]:
```
[(490.7, 0.0028281721954195927)]
```

In [31]:
```
[ n for n in dens if (((ln(n[0])+0.1/Kink)>=lnk) and (ln(n[0])<=lnk)) ]
```

Out[31]:
```
[array([4.9070000e+02, 2.8281722e-03])]
```

In [32]:
```
[(nn[3],(nn[1]/Nobs),nn[1]/Nobs/0.1) for nn in df if nn[3]>(Kink-2) and nn[3]<(K
```

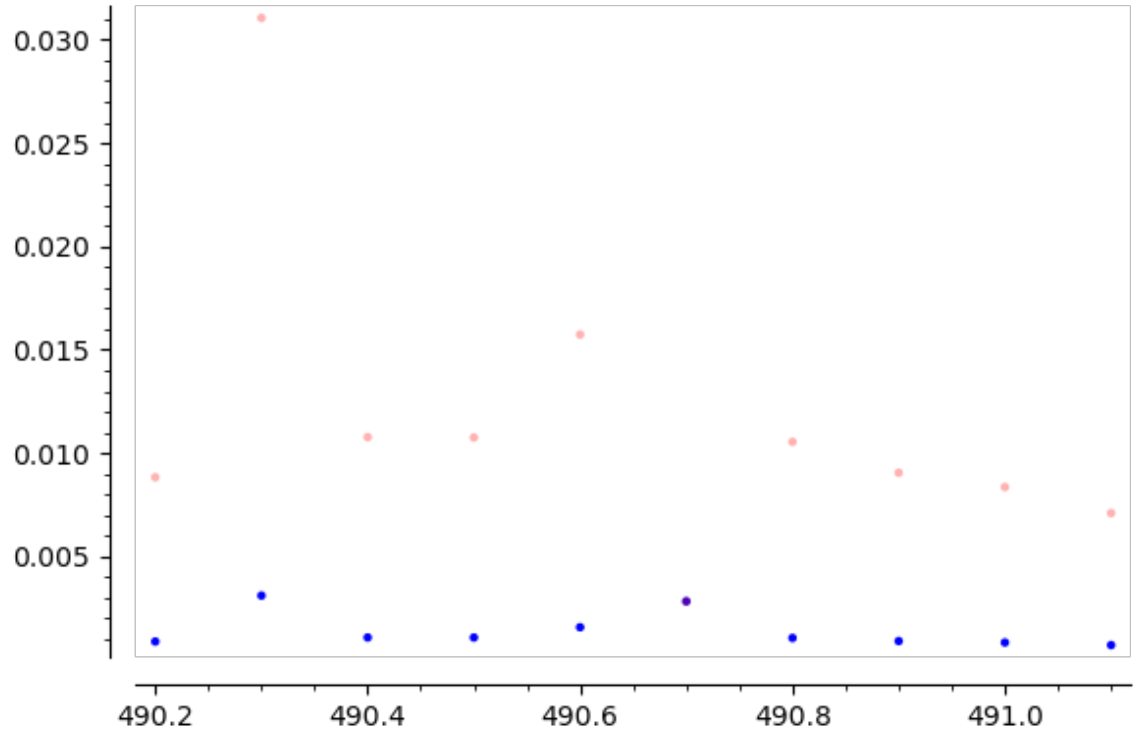
Out[32]:
```
[(488.8, 0.0007646013272038899, 0.007646013272038985),
 (488.9, 0.0007844980160158871, 0.00784498016015887),
 (489.0, 0.0007977624752238851, 0.00797762475223885),
 (489.1, 0.0007939726297358857, 0.00793972629735857),
 (489.2, 0.0007873404001318866, 0.00787340400131866),
 (489.3, 0.0007304927178118948, 0.00730492717811894),
 (489.4, 0.00083281854598788, 0.0083281854598788),
 (489.5, 0.0007987099365958 85, 0.00798709936595885),
 (489.6, 0.0007835505546438872, 0.00783550554643887),
 (489.7, 0.0008309236232438803, 0.00830923623243880),
 (489.8, 0.0008555576189158768, 0.00855557618915877),
 (489.9, 0.0008839814600758727, 0.00883981460075872),
 (490.0, 0.0009038781488878698, 0.00903878148887869),
 (490.1, 0.0009190375308398677, 0.00919037530839867),
 (490.2, 0.0008830339987038728, 0.00883033998703872),
 (490.3, 0.003105778377415553, 0.031057783774155526),
 (490.4, 0.001077263579963845, 0.010772635799638448),
 (490.5, 0.0010753686572198451, 0.01075368657219845),
 (490.6, 0.0015737333388917735, 0.015737333388917734),
 (490.7, 0.0028281721954195927, 0.028281721954195926),
 (490.8, 0.0010545245070358482, 0.010545245070358481),
 (490.9, 0.0009048256102598697, 0.009048256102598696),
 (491.0, 0.0008356609301038797, 0.008356609301038796),
 (491.1, 0.0007096485676278978, 0.007096485676278978),
 (491.2, 0.0007920777069918859, 0.00792077706991886),
 (491.3, 0.0009294596059318662, 0.009294596059318662),
 (491.4, 0.0007844980160158871, 0.00784498016015887),
 (491.5, 0.0007882878615038865, 0.007882878615038865),
 (491.6, 0.0007901827842478862, 0.007901827842478861),
 (491.7, 0.0007826030932718873, 0.007826030932718872),
 (491.8, 0.0007304927178118948, 0.007304927178118948),
 (491.9, 0.0007608114817158904, 0.007608114817158904),
 (492.0, 0.0006556432694239056, 0.006556432694239056),
 (492.1, 0.0007399673315318934, 0.007399673315318934),
 (492.2, 0.0007333351019278944, 0.007333351019278944),
 (492.3, 0.0007854454773878869, 0.007854454773878868),
 (492.4, 0.0007816556318998875, 0.007816556318998875),
 (492.5, 0.0007522843293678917, 0.007522843293678917),
 (492.6, 0.0007494419452518921, 0.00749441945251892)]
```

In [33]:
```
sum([n[1] for n in dens]),sum([n[1] for n in densa])
```

Out[33]:
```
(0.9999999999999999, 9.974546450241231)
```

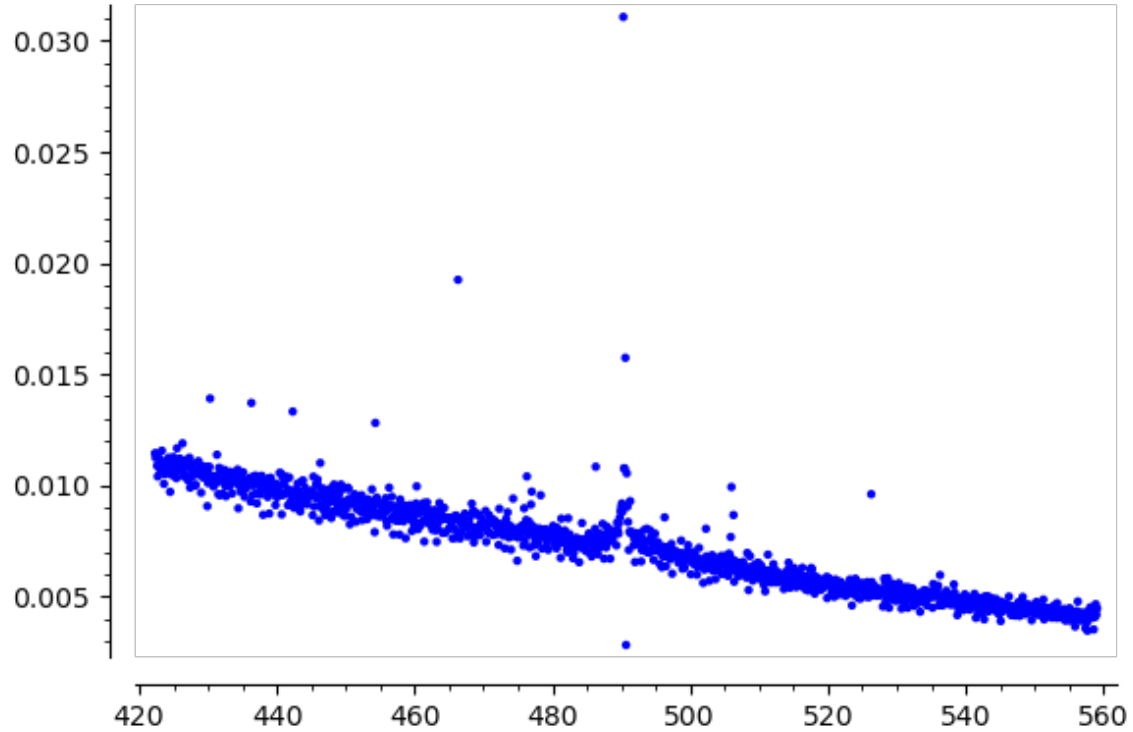
In [34]:
```
list_plot(dens[int(len(dens)/2)-5:int(len(dens)/2)+5])+list_plot(densa[int(len(d
```

Out[34]:




In [35]:
```
list_plot(densa)
```

Out[35]:




In [36]:
```
#sum(dens[n][1] for n in range(0,len(dens)) if n!= KinkPos)+dens[KinkPos][1]
```

In [37]:
```
os.chdir(tt)
tt2=os.getcwd()
show(tt2)
```

# Transformations:

The data is transformed in several ways:

- dens: contains the data as density everywhere but at the kink where it is a probability, used for least squares fitting, model with perfect bunching;
- densa: containes the data as proportion of the sample, used for maximum likelihood, model with perfect bunching;
- densb: contains the data as density everywhere, used for least squares fitting, model with imperfect bunching.

In [38]:
```
totnobs = Nobs

KinkPos = int((len(df)-1)/2)

print(sum( n[1] for n in dens))

#best normal fit...
# measure mean and variance
print("level")
m  = sum(n[1]*n[0]   for n in dens)
m2 = sum(n[1]*n[0]^2 for n in dens)
v2 = m2-m^2
print(['mean',m],['std dev',sqrt(m2-m^2)])

print("logs")
lm = sum( n[1]*log(n[0]) for n in dens)
lm2 = sum(n[1]*log(n[0])^2 for n in dens)
lv2 = lm2-lm^2
print(['mean',lm],['std dev',sqrt(lm2-lm^2)])
# -1. to deal with the data to the left of the lower bound...
# to copy in the spyx file...until I find how to transfer data on the fly!

lnebu = ln(min(n[0] for n in dens)-.1)
lnebo = ln(max(n[0] for n in dens))



sum(dens[n][1] for n in range(0,len(dens)) if n!= KinkPos)+dens[KinkPos][1],\
sum(dens[n][1] for n in range(0,len(dens)) )
```

```
0.9999999999999999
level
['mean', 479.8201495662524] ['std dev', 38.03039478470768]
logs
['mean', 6.170305771954446] ['std dev', 0.07860243742723214]
```

Out[38]: (0.9999999999999999, 0.9999999999999999)

In [39]:
```
lnebu,lnebo
```

Out[39]: (6.0454791354142206, 6.326328348032599)

In [40]:
```
KinkPos,dens[KinkPos]
```

Out[40]: (684, array([4.9070000e+02, 2.8281722e-03]))

In [41]:
```
# allows us to check the data!
#list_plot(dens,ymin=0)+ list_plot(densa,color='gold')
```

In [ ]:
```
```

In [42]:
```
Info_in_year
```

Out[42]:
```
[2019, 0.658500000000000, 0.465850000000000, 490700]
```

In [43]:
```
#list(zip(dens,densa))
Info_in_year[1]
```

Out[43]:
```
0.658500000000000
```

# Tax parameters:

In [44]:
```
pl1,pl2,ps,ps1,ps2,pr12,lw,delta = var('pl1,pl2,ps,ps1,ps2,pr12,lw,delta')
delta = (0.1*.99999)
#/Kink

# this is complicated since their application aggregates many years...here I tak
lnt1c = ln(Info_in_year[1])
lnt2c = ln(Info_in_year[2])
lt1cplt2c = lnt1c^2 + lnt2c^2
lt1cmlt2c = lnt1c^2 - lnt2c^2
```

In [45]:
```
delta,delta/(lnt1c-lnt2c)
```

Out[45]:
```
(0.0999990000000000, 0.288930254180252)
```

In [46]:
```
delta,lnk,dens[KinkPos][0],Kink
```

Out[46]:
```
(0.0999990000000000, 6.19583294309586, 490.7, 490.700000000000)
```

In [47]:
```
#len(df_yy_cnt),Kink
#ck['IG'][0:10],ck['FREQ'][0:10]
#show(list(zip(ck['IG'].list(),ck['FREQ'].list())))
```

# Chetty Polynomial Bunching Estimation, all data

8 different size of the excluded range...

+/- 5
+/- 10
+/- 15
+/- 20
+/- 25
+/- 50
+/- 75
+/- 100

STEP_BIN=0.1

In [48]:
```
load("chetty_bunch.py")
```

In [49]:
```
#bin size (this is more important than you would think!)
STEP_BIN = 0.1
```

In [ ]:
```

```

In [50]:
```
st1 = bin_to_bin(df_yy_cnt,Kink,STEP_BIN,(len(df_yy_cnt)-1)/2)
chetty_tab = []
for exzone in [5,10,15,20,25,50,75,100]:
    # IG, FREQ, list_cnt
    st2 = design_bunch(st1['IG'],st1['FREQ'],Kink,-exzone,exzone,7)
    st3 = chetty_poly_bunch_court(st1['IG'],st1['FREQ'],st2['X'],st2['X_nodum'],
    eresid = [st3['resid'][i][0] for i in range(0,len(st3['y_hat_org'].list()))
    res = lp(st1,st2,st3,eresid,exzone)
    chetty_central_a = st3['b']/(lnt1c-lnt2c)/Kink*STEP_BIN
    to_plot = [a[1]/(lnt1c-lnt2c)/Kink*STEP_BIN for a in res]
    to_plot.sort()
    ltp = len(to_plot)
    chetty_tab.append([chetty_central_a,mean(to_plot),to_plot[round(ltp*.05)],to
```

In [51]:
```
# make a presentable table!
table([map_threaded(lambda x: round(x,3), nn) for nn in list(chetty_tab)],\
      header_row=["Estimate","Bstp-mean","Bstp-5%","Bstp-25%","Bstp-75%","Bstp-9
      header_column=["","Chetty, 5","Chetty, 10","Chetty, 15","Chetty, 20","Chet
```

Out[51]:

| | Estimate | Bstp-mean | Bstp-5% | Bstp-25% | Bstp-75% | Bstp-95% | Bstp-std dev |
|---|---|---|---|---|---|---|---|
| Chetty, 5 | | | | | | | |
| Chetty, 10 | | | | | | | |
| Chetty, 15 | | | | | | | |
| Chetty, 20 | | | | | | | |
| Chetty, 25 | | | | | | | |
| Chetty, 50 | | | | | | | |
| Chetty, 75 | | | | | | | |
| Chetty, 100 | | | | | | | |

In [52]:
```
latex(table([map_threaded(lambda x: round(x,3), nn) for nn in list(chetty_tab)],
      header_row=["Estimate","Bstp-mean","Bstp-5%","Bstp-25%","Bstp-75%","Bstp-9
      header_column=["","Chetty, 5","Chetty, 10","Chetty, 15","Chetty, 20","Chet
```

Out[52]:
```
\begin{tabular}{l|lllllll}
 & Estimate & Bstp-mean & Bstp-5% & Bstp-25% & Bstp-75% & Bstp-95% & Bstp-std de
v \\ \hline
Chetty, 5 & $0.005$ & $0.006$ & $0.005$ & $0.005$ & $0.006$ & $0.006$ & $0.0$ \\
Chetty, 10 & $0.006$ & $0.006$ & $0.006$ & $0.006$ & $0.006$ & $0.007$ & $0.0$
\\
Chetty, 15 & $0.007$ & $0.007$ & $0.007$ & $0.007$ & $0.007$ & $0.007$ & $0.0$
\\
Chetty, 20 & $0.007$ & $0.007$ & $0.007$ & $0.007$ & $0.007$ & $0.007$ & $0.0$
\\
Chetty, 25 & $0.007$ & $0.007$ & $0.007$ & $0.007$ & $0.008$ & $0.008$ & $0.0$
\\
Chetty, 50 & $0.009$ & $0.009$ & $0.008$ & $0.009$ & $0.009$ & $0.01$ & $0.0$ \\
Chetty, 75 & $0.009$ & $0.009$ & $0.008$ & $0.009$ & $0.01$ & $0.01$ & $0.001$
\\
Chetty, 100 & $0.009$ & $0.009$ & $0.007$ & $0.008$ & $0.009$ & $0.01$ & $0.001$
\\
\end{tabular}
```

In [53]:
```
# this would produce some graph...
#chetty_central_a = st3['b']/(lnt1c-lnt2c)/Kink*STEP_BIN
#to_plot = [a[1]/(lnt1c-lnt2c)/Kink*0.1 for a in res]
#mean_to_plot = mean(to_plot)
#sdv_to_plot = sqrt(variance(to_plot,bias=True))
#show([mean_to_plot,sdv_to_plot])
#P = plot(pdfn((x-mean_to_plot)/sdv_to_plot)/sdv_to_plot, (x, mean_to_plot-5.*sd
#L_ch = line([(chetty_central_a,0),(chetty_central_a,1.2*pdfn(0.)/sdv_to_plot)],
#histogram(to_plot,bins=131,density=True)+P+L_ch
```

In [ ]:

In [ ]:

# Chetty Polynomial Bunching Estimation, 450.7-530.7

8 different size of the excluded range...

+/- 5
+/- 10
+/- 15
+/- 20
+/- 25
+/- 50
+/- 75
+/- 100

STEP_BIN=0.1

In [54]:
```
load("chetty_bunch.py")
```

In [55]:
```
#bin size (this is more important than you would think!)
STEP_BIN = 0.1
```

In [ ]:

In [56]:
```
#data selection
topdistfromk = 40+0.001


df = [nn for nn in dd if abs(nn[3]-Kink)<=topdistfromk ]
df_yy_cnt = [ (dd[3],dd[1]) for dd in df]
Nobs = int(sum([nn[1] for nn in df]))
show(Nobs)
show(len(df_yy_cnt))
```

In [57]:
```
st1 = bin_to_bin(df_yy_cnt,Kink,STEP_BIN,(len(df_yy_cnt)-1)/2)
chetty_tab = []
for exzone in [5,10,15,20,25,50,75,100]:
    # IG, FREQ, list_cnt
    st2 = design_bunch(st1['IG'],st1['FREQ'],Kink,-exzone,exzone,7)
    st3 = chetty_poly_bunch_court(st1['IG'],st1['FREQ'],st2['X'],st2['X_nodum'],
    eresid = [st3['resid'][i][0] for i in range(0,len(st3['y_hat_org'].list()))
    res = lp(st1,st2,st3,eresid,exzone)
    chetty_central_a = st3['b']/(lnt1c-lnt2c)/Kink*STEP_BIN
    to_plot = [a[1]/(lnt1c-lnt2c)/Kink*STEP_BIN for a in res]
    to_plot.sort()
    ltp = len(to_plot)
    chetty_tab.append([chetty_central_a,mean(to_plot),to_plot[round(ltp*.05)],to
```

In [58]:
```
# make a presentable table!
table([map_threaded(lambda x: round(x,3), nn) for nn in list(chetty_tab)],\
      header_row=["Estimate","Bstp-mean","Bstp-5%","Bstp-25%","Bstp-75%","Bstp-9
      header_column=["","Chetty, 5","Chetty, 10","Chetty, 15","Chetty, 20","Chet
```

Out[58]:

| | Estimate | Bstp-mean | Bstp-5% | Bstp-25% | Bstp-75% | Bstp-95% | Bstp-std dev |
|---|---|---|---|---|---|---|---|
| Chetty, 5 | | | | | | | |
| Chetty, 10 | | | | | | | |
| Chetty, 15 | | | | | | | |
| Chetty, 20 | | | | | | | |
| Chetty, 25 | | | | | | | |
| Chetty, 50 | | | | | | | |
| Chetty, 75 | | | | | | | |
| Chetty, 100 | | | | | | | |

In [59]:
```
latex(table([map_threaded(lambda x: round(x,3), nn) for nn in list(chetty_tab)],
      header_row=["Estimate","Bstp-mean","Bstp-5%","Bstp-25%","Bstp-75%","Bstp-9
      header_column=["","Chetty, 5","Chetty, 10","Chetty, 15","Chetty, 20","Chet
```

Out[59]:
```
\begin{tabular}{l|lllllll}
 & Estimate & Bstp-mean & Bstp-5% & Bstp-25% & Bstp-75% & Bstp-95% & Bstp-std de
v \\ \hline
Chetty, 5 & $0.005$ & $0.005$ & $0.005$ & $0.005$ & $0.005$ & $0.006$ & $0.0$ \\
Chetty, 10 & $0.006$ & $0.006$ & $0.006$ & $0.006$ & $0.006$ & $0.006$ & $0.0$
\\
Chetty, 15 & $0.006$ & $0.007$ & $0.006$ & $0.006$ & $0.007$ & $0.007$ & $0.0$
\\
Chetty, 20 & $0.007$ & $0.007$ & $0.007$ & $0.007$ & $0.007$ & $0.007$ & $0.0$
\\
Chetty, 25 & $0.007$ & $0.007$ & $0.007$ & $0.007$ & $0.007$ & $0.008$ & $0.0$
\\
Chetty, 50 & $0.009$ & $0.009$ & $0.008$ & $0.009$ & $0.01$ & $0.01$ & $0.001$
\\
Chetty, 75 & $0.009$ & $0.009$ & $0.007$ & $0.009$ & $0.01$ & $0.011$ & $0.001$
\\
Chetty, 100 & $0.008$ & $0.008$ & $0.005$ & $0.007$ & $0.009$ & $0.011$ & $0.00
2$ \\
\end{tabular}
```

In [60]:
```
# this would produce some graph...
#chetty_central_a = st3['b']/(lnt1c-lnt2c)/Kink*STEP_BIN
#to_plot = [a[1]/(lnt1c-lnt2c)/Kink*0.1 for a in res]
#mean_to_plot = mean(to_plot)
#sdv_to_plot = sqrt(variance(to_plot,bias=True))
#show([mean_to_plot,sdv_to_plot])
#P = plot(pdfn((x-mean_to_plot)/sdv_to_plot)/sdv_to_plot, (x, mean_to_plot-5.*sd
#L_ch = line([(chetty_central_a,0),(chetty_central_a,1.2*pdfn(0.)/sdv_to_plot)],
#histogram(to_plot,bins=131,density=True)+P+L_ch
```

In [ ]:
```
```

# Chetty Polynomial Bunching Estimation, 475.7-505.7

8 different size of the excluded range...

+/- 5

+/- 10

+/- 15

+/- 20

+/- 25

+/- 50

+/- 75

+/- 100

STEP_BIN=0.1

In [61]:
```
load("chetty_bunch.py")
```

In [62]:
```
#bin size (this is more important than you would think!)
STEP_BIN = 0.1
```

In [63]:
```
#data selection
topdistfromk = 15+0.001
df = [nn for nn in dd if abs(nn[3]-Kink)<=topdistfromk ]
df_yy_cnt = [ (dd[3],dd[1]) for dd in df]
Nobs = int(sum([nn[1] for nn in df]))
show(Nobs)
show(len(df_yy_cnt))
```

In [64]:
```
st1 = bin_to_bin(df_yy_cnt,Kink,STEP_BIN,(len(df_yy_cnt)-1)/2)
chetty_tab = []
for exzone in [5,10,15,20,25,50,75,100]:
    # IG, FREQ, list_cnt
    st2 = design_bunch(st1['IG'],st1['FREQ'],Kink,-exzone,exzone,7)
    st3 = chetty_poly_bunch_court(st1['IG'],st1['FREQ'],st2['X'],st2['X_nodum'],
    eresid = [st3['resid'][i][0] for i in range(0,len(st3['y_hat_org'].list()))
    res = lp(st1,st2,st3,eresid,exzone)
    chetty_central_a = st3['b']/(lnt1c-lnt2c)/Kink*STEP_BIN
    to_plot = [a[1]/(lnt1c-lnt2c)/Kink*STEP_BIN for a in res]
    to_plot.sort()
    ltp = len(to_plot)
    chetty_tab.append([chetty_central_a,mean(to_plot),to_plot[round(ltp*.05)],to
```

In [65]:
```
# make a presentable table!
table([map_threaded(lambda x: round(x,3), nn) for nn in list(chetty_tab)],\
      header_row=["Estimate","Bstp-mean","Bstp-5%","Bstp-25%","Bstp-75%","Bstp-9
      header_column=["","Chetty, 5","Chetty, 10","Chetty, 15","Chetty, 20","Chet
```

Out[65]:

| | Estimate | Bstp-mean | Bstp-5% | Bstp-25% | Bstp-75% | Bstp-95% | Bstp-std dev |
|---|---|---|---|---|---|---|---|
| Chetty, 5 | | | | | | | |
| Chetty, 10 | | | | | | | |
| Chetty, 15 | | | | | | | |
| Chetty, 20 | | | | | | | |
| Chetty, 25 | | | | | | | |
| Chetty, 50 | | | | | | | |
| Chetty, 75 | | | | | | | |
| Chetty, 100 | | | | | | | |

In [66]:
```
latex(table([map_threaded(lambda x: round(x,3), nn) for nn in list(chetty_tab)],
      header_row=["Estimate","Bstp-mean","Bstp-5%","Bstp-25%","Bstp-75%","Bstp-9
      header_column=["","Chetty, 5","Chetty, 10","Chetty, 15","Chetty, 20","Chet
```

Out[66]:
```
\begin{tabular}{l|lllllll}
 & Estimate & Bstp-mean & Bstp-5% & Bstp-25% & Bstp-75% & Bstp-95% & Bstp-std de
v \\ \hline
Chetty, 5 & $0.005$ & $0.005$ & $0.005$ & $0.005$ & $0.005$ & $0.005$ & $0.0$ \\
Chetty, 10 & $0.006$ & $0.006$ & $0.005$ & $0.005$ & $0.006$ & $0.006$ & $0.0$
\\
Chetty, 15 & $0.006$ & $0.006$ & $0.005$ & $0.005$ & $0.006$ & $0.006$ & $0.0$
\\
Chetty, 20 & $0.006$ & $0.006$ & $0.005$ & $0.006$ & $0.006$ & $0.006$ & $0.0$
\\
Chetty, 25 & $0.006$ & $0.006$ & $0.005$ & $0.005$ & $0.006$ & $0.006$ & $0.0$
\\
Chetty, 50 & $0.009$ & $0.009$ & $0.006$ & $0.008$ & $0.01$ & $0.012$ & $0.002$
\\
Chetty, 75 & $0.014$ & $0.015$ & $0.003$ & $0.009$ & $0.02$ & $0.028$ & $0.008$
\\
Chetty, 100 & $-0.003$ & $0.007$ & $-0.039$ & $-0.021$ & $0.022$ & $0.09$ & $0.0
92$ \\
\end{tabular}
```

In [67]:
```
# this would produce some graph...
#chetty_central_a = st3['b']/(lnt1c-lnt2c)/Kink*STEP_BIN
#to_plot = [a[1]/(lnt1c-lnt2c)/Kink*0.1 for a in res]
#mean_to_plot = mean(to_plot)
#sdv_to_plot = sqrt(variance(to_plot,bias=True))
#show([mean_to_plot,sdv_to_plot])
#P = plot(pdfn((x-mean_to_plot)/sdv_to_plot)/sdv_to_plot, (x, mean_to_plot-5.*sd
#L_ch = line([(chetty_central_a,0),(chetty_central_a,1.2*pdfn(0.)/sdv_to_plot)],
#histogram(to_plot,bins=131,density=True)+P+L_ch
```