0.1 Frequency Keeping data...

The frequency keeping data has been made avialable on-line since September 2005.
Published on the EC/EA's website and provided in the form of monthly excel spreadsheets from the System Operator.

There are two types of files of FK data: Offers, contained in files usually titled something like, FK_Cleared_Offers files, and, Constrained on and off costs contained in files usually titled like, FK_Constrained_Costs The FK_Cleared_Offers gives, per trading period, the winning FK offer price, along with other info such as the winning trader. The FK_Constrained_Costs gives the constrained on and off costs for each trading period.

The data is *not* pretty and there are plenty of issues, as discussed. You could say it is a complete *mess*, but on the bright-side, at least we have *some* data... as an aside, are there any obligations to provide this data in the Code?

For data since 1 Jan, 2010, a script was written to download FK data from https://www.ea.govt.nz/industry/pso-cq/system-operations/fk-data/. Previous data was grabbed from emails (I kid you not), which were saved in FileSite (the EA's document management system). All attachments from these various emails were saved into a single directory of raw data which was then split into years.

Different file names, column names, even different file types..., some data was missing (April 2007) but found on the EC archival website. Yay. The cleared offers file for Dec 2008 was interesting as it appeared to have been saved, not as an xls file, but as an xlm file! Managed to fix this by downloading the EC archival copy and coping and pasting each sheet into a new workbook...

Numerous other inconsistancies have been incounted such as data which should not have been present, being present, and also missing data.

Currently there are two FK markets, one for each island. For each market there has historically been two participants. Originally there was CTCT and MERI in the South Island, and MRPL and GENE in the North.

In 2012 CTCT also started bidding into the frequency keeping market in the North Island, so there are now two parties in the South and three in the North.

This requires special mapping (as demonstrated in this notebook)...

And then problems with dates.... I sound like Nicky.

Data from October 2007 was particularly problamatic. After many failed attempts in Excel, this required it's own hack, to get date formats to comply! MS Excel date formating is one ugly beast...

Of course, the filenames were not consistent, so these were renamed with a bit of python magic into a consistant order, YYYY_MM_XXXXXXX.xls, see some of the scripts used at the end of this notebook.

This notebook contains most of the code used to tidy up this historic data set.

D.J.Hume, 18th March, 2013.

0.1.1 Import required modules

```
from pandas import *
from pandas.util.testing import set_trace as st
from gnash import *
import numpy as np
from bs4 import BeautifulSoup
```

```
import mechanize
import cookielib
import os,sys
from datetime import date, datetime, time, timedelta
from io import StringIO
import logging
import logging.handlers
import time as tm
set_option('max_rows',1000)
```

```
%config InlineBackend.figure_format='png'
```

0.1.2 Set path and create file lists

```
path = '/home/humed/python/fkdata/'
os.chdir(path)

fl=[]
for dirn, dirns, f in os.walk(path+'data'):
    for filex in f:
        fl.append(dirn + '/' + filex)
fl.sort() #print os.path.join(dirn,f)
fl_co = Series(fl)[Series(fl).map(lambda x: x.split('_')[2][:2]=='co')].values
fl_cl = Series(fl)[Series(fl).map(lambda x: x.split('_')[2][:2]=='cl')].values
```

0.1.3 Function to parse and merge the data

```
def parse_data(filelist,offers_or_cost,block_island_map):

    def island_split(df,offers_or_cost):
        if offers_or_cost == 'clearedoffers':
            NI = df[df['block or market node'].map(lambda x: block_island_map[x]=='NI')]
            SI = df[df['block or market node'].map(lambda x: block_island_map[x]=='SI')]
        if offers_or_cost == 'constrainedcosts':
            NI = df[df['GIP_GXP_FULL'].map(lambda x: block_island_map[x]=='NI')]
            SI = df[df['GIP_GXP_FULL'].map(lambda x: block_island_map[x]=='SI')]

        return NI,SI

    def mult_merge(df_list,merge_on):
        if not df_list[0].empty:
            df=df_list[0]
        for i in range(len(df_list)-1):
            if not df_list[i+1].empty:
                df=merge(df,df_list[i+1],on=merge_on,left_index=True,right_index=True,how=
                    'outer')
        return df

    comps = ['CTCT','GENE','MRPL','MERI']
```

```
    if offers_or_cost == 'constrainedcosts':
        parse_cols=[0,1,2,3,4,5,6,7,8,9]
        #parse_cols=[]
        index_set = ['TRADING_DATE','TRADING_PERIOD']
        merge_on_2=['ORG_CODE','GIP_GXP_FULL','QUANTITY','PRICE','AMT','CON_IND']
        merge_on=['ORG_CODE','GIP_GXP_FULL','QUANTITY','PRICE','AMOUNT','CON_IND']
        skiprows=[0,1,2,3,4]
    if offers_or_cost == 'clearedoffers':
        parse_cols=[0,1,2,3,4,5,6]
        index_set = ['trading date','trading period']
        merge_on=['trader','block or market node','frequency qty','partperiod','offer']
        merge_on_2=['trader','block or market node','frequency qty','partperiod','offer']
        skiprows=''
    for i,f in enumerate(filelist):
        #print i,f
        (title, extn) = os.path.splitext(f)
        if extn==".xls" or extn==".xlsx":
            if offers_or_cost in title:
                print f
                xls_file = ExcelFile(f)
                #get all data per trader
                #alld = [parsedf(c) for c in comps]

                CTCT = xls_file.parse('CTCT',parse_dates=True,parse_cols=parse_cols,
                    skiprows=skiprows).set_index(index_set).dropna(how='all')
                MERI = xls_file.parse('MERI',parse_dates=True,parse_cols=parse_cols,
                    skiprows=skiprows).set_index(index_set).dropna(how='all')
                GENE = xls_file.parse('GENE',parse_dates=True,parse_cols=parse_cols,
                    skiprows=skiprows).set_index(index_set).dropna(how='all')
                MRPL = xls_file.parse('MRPL',parse_dates=True,parse_cols=parse_cols,
                    skiprows=skiprows).set_index(index_set).dropna(how='all')
                #now split into islands

                [CTCT_NI,CTCT_SI]=island_split(CTCT,offers_or_cost)
                [MERI_NI,MERI_SI]=island_split(MERI,offers_or_cost)
                [GENE_NI,GENE_SI]=island_split(GENE,offers_or_cost)
                [MRPL_NI,MRPL_SI]=island_split(MRPL,offers_or_cost)

                if i==0:
                    #print i
                    SI=mult_merge([MERI_SI,CTCT_SI,GENE_SI,MRPL_SI],merge_on)
                    NI=mult_merge([GENE_NI,MRPL_NI,CTCT_NI,MERI_NI],merge_on)

                if i>0:
                    try:
                        SI=SI.append(mult_merge([MERI_SI,CTCT_SI,GENE_SI,MRPL_SI],merge_on)
                            )
                        NI=NI.append(mult_merge([MRPL_NI,GENE_NI,CTCT_NI,MERI_NI],merge_on)
                            )
                    except:
                        SI=SI.append(mult_merge([MERI_SI,CTCT_SI,GENE_SI,MRPL_SI],
```

```
                        merge_on_2))
                    NI=NI.append(mult_merge([MRPL_NI,GENE_NI,CTCT_NI,MERI_NI],
                        merge_on_2))
                    continue

    return NI,SI
```

0.1.4 Sort out useful mappings for this data set

```
#Get unique frequency keeping block ids for 2012
islands = ['SI','SI','NI','NI','SI','NI','NI','NI','NI','NI','NI','NI']
blocks = []
all_offers = read_csv('/home/humed/python/fkdata/raw/finalFK2012.csv')
for pn in unique(all_offers['PNode Name']):
    #print pn.split(' ')[0].strip()
    blocks.append(pn)
block_island_map=dict(zip(blocks,islands))
block_island_map['HLY2201 HLY4']='NI' #add HLY4 for previous years (prior to 2012)
    otherwise key error
block_island_map
```

```
{'CLU': 'SI',
 'HLY2201 HLY1': 'NI',
 'HLY2201 HLY2': 'NI',
 'HLY2201 HLY3': 'NI',
 'HLY2201 HLY4': 'NI',
 'MAN': 'SI',
 'SFD2201 SFD21': 'NI',
 'SFD2201 SFD22': 'NI',
 'SFD2201 SPL0': 'NI',
 'TKU2201 TKU0': 'NI',
 'WKA': 'NI',
 'WTO': 'NI',
 'WTR': 'SI'}
```

block$_{i}slandmapforcleareddata$

```
#loop though all files to get the unique GIPs
#by ruturning the dict we were able to pick up a few more data errors... nans return as
    gips...
from itertools import chain
all_gips=[]
all_gips2={}

for f in fl_co:
    xls_file = ExcelFile(f)
    parse_cols=[0,1,2,3,4,5,6,7,8,9]
    index_set = ['TRADING_DATE','TRADING_PERIOD']
    merge_on_2=['ORG_CODE','GIP_GXP_FULL','QUANTITY','PRICE','AMT','CON_IND','STATION']
    merge_on=['ORG_CODE','GIP_GXP_FULL','QUANTITY','PRICE','AMOUNT','CON_IND','STATION']
```

```
    skiprows=[0,1,2,3,4]


    CTCT = xls_file.parse('CTCT',parse_dates=True,parse_cols=parse_cols,skiprows=skiprows
        ).set_index(index_set).dropna(how='all')
    MERI = xls_file.parse('MERI',parse_dates=True,parse_cols=parse_cols,skiprows=skiprows
        ).set_index(index_set).dropna(how='all')
    GENE = xls_file.parse('GENE',parse_dates=True,parse_cols=parse_cols,skiprows=skiprows
        ).set_index(index_set).dropna(how='all')
    MRPL = xls_file.parse('MRPL',parse_dates=True,parse_cols=parse_cols,skiprows=skiprows
        ).set_index(index_set).dropna(how='all')

    all_gips.append(list(unique(CTCT.GIP_GXP_FULL)))
    all_gips.append(list(unique(MERI.GIP_GXP_FULL)))
    all_gips.append(list(unique(GENE.GIP_GXP_FULL)))
    all_gips.append(list(unique(MRPL.GIP_GXP_FULL)))

    x = [list(unique(CTCT.GIP_GXP_FULL))]
    x.append(list(unique(MERI.GIP_GXP_FULL)))
    x.append(list(unique(GENE.GIP_GXP_FULL)))
    x.append(list(unique(MRPL.GIP_GXP_FULL)))
    all_gips2[f]=unique(list(chain.from_iterable(x)))


GIPs = unique(list(chain.from_iterable(all_gips2.values())))
islandsC = ['SI','SI','NI','NI','NI','NI','NI']
block_island_map2 = dict(zip(GIPs,islandsC))
block_island_map2
```

```
{u'CLU': 'SI',
 u'HLY': 'NI',
 u'SFD': 'NI',
 u'TRO': 'NI',
 u'WKA': 'NI',
 u'WTO': 'NI',
 u'WTR': 'SI'}
```

```
#for each file, look at the block GIPs /check.
all_gips2
```

```
{'/home/humed/python/fkdata/data/2005/2005_09_constrainedcosts.xls': array([CLU, WTR, HLY, WKA, TRO, WTO
 '/home/humed/python/fkdata/data/2005/2005_10_constrainedcosts.xls': array([CLU, WTR, TRO, HLY, WTO], dt
 '/home/humed/python/fkdata/data/2005/2005_11_constrainedcosts.xls': array([CLU, WTR, HLY, TRO, WTO], dt
 '/home/humed/python/fkdata/data/2005/2005_12_constrainedcosts.xls': array([CLU, WTR, HLY, TRO, WKA, WTO
 '/home/humed/python/fkdata/data/2006/2006_01_constrainedcosts.xls': array([CLU, WTR, HLY, TRO, WKA, WTO
 '/home/humed/python/fkdata/data/2006/2006_02_constrainedcosts.xls': array([CLU, WTR, WKA, HLY, TRO, WTO
 '/home/humed/python/fkdata/data/2006/2006_03_constrainedcosts.xls': array([CLU, WTR, HLY, WKA, TRO, WTO
 '/home/humed/python/fkdata/data/2006/2006_04_constrainedcosts.xls': array([CLU, WTR, HLY, WKA, TRO, WTO
 '/home/humed/python/fkdata/data/2006/2006_05_constrainedcosts.xls': array([CLU, WTR, TRO, WKA, HLY, WTO
 '/home/humed/python/fkdata/data/2006/2006_06_constrainedcosts.xls': array([CLU, WTR, WKA, TRO, HLY, WTO
 '/home/humed/python/fkdata/data/2006/2006_07_constrainedcosts.xls': array([CLU, WTR, HLY, TRO, WKA, WTO
```

```
'/home/humed/python/fkdata/data/2006/2006_08_constrainedcosts.xls': array([CLU, WTR, WKA, TRO, HLY, WTO
'/home/humed/python/fkdata/data/2006/2006_09_constrainedcosts.xls': array([CLU, WTR, WKA, HLY, TRO, WTO
'/home/humed/python/fkdata/data/2006/2006_10_constrainedcosts.xls': array([CLU, WTR, HLY, TRO, WKA, WTO
'/home/humed/python/fkdata/data/2006/2006_11_constrainedcosts.xls': array([CLU, WTR, TRO, HLY, WTO], dt
'/home/humed/python/fkdata/data/2006/2006_12_constrainedcosts.xls': array([CLU, WTR, TRO, WTO], dtype=o
'/home/humed/python/fkdata/data/2007/2007_01_constrainedcosts.xls': array([CLU, WTR, TRO, WKA, WTO], dt
'/home/humed/python/fkdata/data/2007/2007_02_constrainedcosts.xls': array([CLU, WTR, TRO, WKA, WTO], dt
'/home/humed/python/fkdata/data/2007/2007_03_constrainedcosts.xls': array([CLU, WTR, WKA, TRO, WTO], dt
'/home/humed/python/fkdata/data/2007/2007_04_constrainedcosts.xls': array([WTR, WKA, TRO, WTO], dtype=o
'/home/humed/python/fkdata/data/2007/2007_05_constrainedcosts.xls': array([CLU, WTR, TRO, WKA, WTO], dt
'/home/humed/python/fkdata/data/2007/2007_06_constrainedcosts.xls': array([CLU, WTR, TRO, WKA, WTO], dt
'/home/humed/python/fkdata/data/2007/2007_07_constrainedcosts.xls': array([CLU, WTR, TRO, WTO], dtype=o
'/home/humed/python/fkdata/data/2007/2007_08_constrainedcosts.xls': array([CLU, WTR, TRO, WKA, WTO], dt
'/home/humed/python/fkdata/data/2007/2007_09_constrainedcosts.xls': array([CLU, WTR, TRO, WTO], dtype=o
'/home/humed/python/fkdata/data/2007/2007_10_constrainedcosts.xls': array([CLU, WTR, TRO, WTO], dtype=o
'/home/humed/python/fkdata/data/2007/2007_11_constrainedcosts.xls': array([CLU, WTR, TRO, WTO], dtype=o
'/home/humed/python/fkdata/data/2007/2007_12_constrainedcosts.xls': array([CLU, WTR, TRO, WTO], dtype=o
'/home/humed/python/fkdata/data/2008/2008_01_constrainedcosts.xls': array([CLU, WTR, TRO, WKA, WTO], dt
'/home/humed/python/fkdata/data/2008/2008_02_constrainedcosts.xls': array([CLU, WTR, WKA, TRO, WTO], dt
'/home/humed/python/fkdata/data/2008/2008_03_constrainedcosts.xls': array([CLU, WTR, WKA, TRO, WTO], dt
'/home/humed/python/fkdata/data/2008/2008_04_constrainedcosts.xls': array([CLU, WTR, WKA, TRO, WTO], dt
'/home/humed/python/fkdata/data/2008/2008_05_constrainedcosts.xls': array([CLU, WTR, TRO, WKA, WTO], dt
'/home/humed/python/fkdata/data/2008/2008_06_constrainedcosts.xls': array([CLU, WTR, WKA, TRO, WTO], dt
'/home/humed/python/fkdata/data/2008/2008_07_constrainedcosts.xls': array([CLU, WTR, WKA, TRO, WTO], dt
'/home/humed/python/fkdata/data/2008/2008_08_constrainedcosts.xls': array([CLU, WTR, WKA, TRO, HLY, WTO
'/home/humed/python/fkdata/data/2008/2008_09_constrainedcosts.xls': array([CLU, WTR, WKA, TRO, WTO], dt
'/home/humed/python/fkdata/data/2008/2008_10_constrainedcosts.xls': array([CLU, WTR, WKA, TRO, HLY, WTO
'/home/humed/python/fkdata/data/2008/2008_11_constrainedcosts.xls': array([CLU, WTR, HLY, TRO, WTO], dt
'/home/humed/python/fkdata/data/2008/2008_12_constrainedcosts.xls': array([CLU, WTR, HLY, TRO, WTO], dt
'/home/humed/python/fkdata/data/2009/2009_01_constrainedcosts.xls': array([CLU, WTR, TRO, HLY, WTO], dt
'/home/humed/python/fkdata/data/2009/2009_02_constrainedcosts.xls': array([CLU, WTR, TRO, HLY, WTO], dt
'/home/humed/python/fkdata/data/2009/2009_03_constrainedcosts.xls': array([CLU, WTR, TRO, HLY, WKA, WTO
'/home/humed/python/fkdata/data/2009/2009_04_constrainedcosts.xls': array([CLU, WTR, TRO, HLY, WKA, WTO
'/home/humed/python/fkdata/data/2009/2009_05_constrainedcosts.xls': array([CLU, WTR, TRO, HLY, WKA, WTO
'/home/humed/python/fkdata/data/2009/2009_06_constrainedcosts.xls': array([CLU, WTR, HLY, TRO, WKA, WTO
'/home/humed/python/fkdata/data/2009/2009_07_constrainedcosts.xls': array([CLU, WTR, WKA, HLY, TRO, WTO
'/home/humed/python/fkdata/data/2009/2009_08_constrainedcosts.xls': array([CLU, WTR, TRO, HLY, WKA, WTO
'/home/humed/python/fkdata/data/2009/2009_09_constrainedcosts.xls': array([CLU, WTR, TRO, WKA, WTO], dt
'/home/humed/python/fkdata/data/2009/2009_10_constrainedcosts.xls': array([CLU, WTR, TRO, WKA, WTO], dt
'/home/humed/python/fkdata/data/2009/2009_11_constrainedcosts.xls': array([CLU, WTR, WKA, TRO, HLY, WTO
'/home/humed/python/fkdata/data/2009/2009_12_constrainedcosts.xls': array([CLU, WTR, TRO, WTO], dtype=o
'/home/humed/python/fkdata/data/2010/2010_01_constrainedcosts.xls': array([CLU, WTR, TRO, HLY, WTO], dt
'/home/humed/python/fkdata/data/2010/2010_02_constrainedcosts.xls': array([CLU, WTR, TRO, HLY, WKA, WTO
'/home/humed/python/fkdata/data/2010/2010_03_constrainedcosts.xls': array([CLU, WTR, WKA, HLY, TRO, WTO
'/home/humed/python/fkdata/data/2010/2010_04_constrainedcosts.xls': array([CLU, WTR, WKA, TRO, HLY, WTO
'/home/humed/python/fkdata/data/2010/2010_05_constrainedcosts.xls': array([CLU, WTR, WKA, HLY, TRO, WTO
'/home/humed/python/fkdata/data/2010/2010_06_constrainedcosts.xls': array([CLU, WTR, HLY, TRO, WKA, WTO
'/home/humed/python/fkdata/data/2010/2010_07_constrainedcosts.xls': array([CLU, WTR, WKA, TRO, HLY, WTO
'/home/humed/python/fkdata/data/2010/2010_08_constrainedcosts.xls': array([CLU, WTR, TRO, WKA, HLY, WTO
'/home/humed/python/fkdata/data/2010/2010_09_constrainedcosts.xls': array([CLU, WTR, TRO, WKA, HLY, WTO
'/home/humed/python/fkdata/data/2010/2010_10_constrainedcosts.xls': array([CLU, WTR, WKA, TRO, WTO], dt
'/home/humed/python/fkdata/data/2010/2010_11_constrainedcosts.xls': array([CLU, WTR, TRO, HLY, WKA, WTO
```

```
'/home/humed/python/fkdata/data/2010/2010_12_constrainedcosts.xls': array([CLU, WTR, TRO, HLY, WKA, WTO
'/home/humed/python/fkdata/data/2011/2011_01_constrainedcosts.xls': array([CLU, WTR, TRO, WKA, HLY, WTO
'/home/humed/python/fkdata/data/2011/2011_02_constrainedcosts.xls': array([CLU, WTR, TRO, HLY, WKA, WTO
'/home/humed/python/fkdata/data/2011/2011_03_constrainedcosts.xls': array([CLU, WTR, WKA, TRO, WTO], dt
'/home/humed/python/fkdata/data/2011/2011_04_constrainedcosts.xls': array([CLU, WTR, TRO, WKA, HLY, WTO
'/home/humed/python/fkdata/data/2011/2011_05_constrainedcosts.xls': array([CLU, WTR, WKA, TRO, WTO], dt
'/home/humed/python/fkdata/data/2011/2011_06_constrainedcosts.xls': array([CLU, WTR, TRO, WKA, WTO], dt
'/home/humed/python/fkdata/data/2011/2011_07_constrainedcosts.xls': array([CLU, SFD, WTR, TRO, WKA, HLY
'/home/humed/python/fkdata/data/2011/2011_08_constrainedcosts.xls': array([CLU, WTR, WKA, TRO, WTO], dt
'/home/humed/python/fkdata/data/2011/2011_09_constrainedcosts.xls': array([CLU, WTR, WKA, TRO, HLY, WTO
'/home/humed/python/fkdata/data/2011/2011_10_constrainedcosts.xls': array([CLU, WTR, HLY, TRO, WKA, WTO
'/home/humed/python/fkdata/data/2011/2011_11_constrainedcosts.xls': array([CLU, WTR, TRO, WKA, WTO], dt
'/home/humed/python/fkdata/data/2011/2011_12_constrainedcosts.xls': array([CLU, SFD, WTR, TRO, WKA, WTO
'/home/humed/python/fkdata/data/2012/2012_01_constrainedcosts.xls': array([CLU, SFD, WTR, TRO, WKA, WTO
'/home/humed/python/fkdata/data/2012/2012_02_constrainedcosts.xls': array([CLU, SFD, WTR, TRO, WKA, WTO
'/home/humed/python/fkdata/data/2012/2012_03_constrainedcosts.xls': array([CLU, SFD, WTR, TRO, WKA, HLY
'/home/humed/python/fkdata/data/2012/2012_04_constrainedcosts.xls': array([CLU, SFD, WTR, WKA, TRO, WTO
'/home/humed/python/fkdata/data/2012/2012_05_constrainedcosts.xls': array([CLU, SFD, WTR, WKA, TRO, HLY
'/home/humed/python/fkdata/data/2012/2012_06_constrainedcosts.xls': array([CLU, SFD, WTR, WKA, TRO, WTO
'/home/humed/python/fkdata/data/2012/2012_07_constrainedcosts.xls': array([CLU, SFD, WTR, TRO, WKA, WTO
'/home/humed/python/fkdata/data/2012/2012_08_constrainedcosts.xls': array([CLU, SFD, WTR, TRO, WKA, WTO
'/home/humed/python/fkdata/data/2012/2012_09_constrainedcosts.xls': array([CLU, SFD, WTR, TRO, WKA, WTO
'/home/humed/python/fkdata/data/2012/2012_10_constrainedcosts.xls': array([CLU, SFD, WTR, TRO, WKA, WTO
'/home/humed/python/fkdata/data/2012/2012_11_constrainedcosts.xls': array([CLU, SFD, WTR, TRO, WKA, WTO
'/home/humed/python/fkdata/data/2012/2012_12_constrainedcosts.xls': array([SFD, CLU, WTR, TRO, WTO], dt
'/home/humed/python/fkdata/data/2013/2013_01_constrainedcosts.xls': array([CLU, SFD, WTR, TRO, WKA, WTO
```

block island map for constrained cost data

0.1.5 Parse the cleared offers data

```
[NI,SI] = parse_data(fl_cl,'clearedoffers',block_island_map)
```

```
/home/humed/python/fkdata/data/2005/2005_09_clearedoffers.xls
/home/humed/python/fkdata/data/2005/2005_10_clearedoffers.xls

/home/humed/python/fkdata/data/2005/2005_11_clearedoffers.xls

/home/humed/python/fkdata/data/2005/2005_12_clearedoffers.xls

/home/humed/python/fkdata/data/2006/2006_01_clearedoffers.xls

/home/humed/python/fkdata/data/2006/2006_02_clearedoffers.xls

/home/humed/python/fkdata/data/2006/2006_03_clearedoffers.xls

/home/humed/python/fkdata/data/2006/2006_04_clearedoffers.xls

/home/humed/python/fkdata/data/2006/2006_05_clearedoffers.xls

/home/humed/python/fkdata/data/2006/2006_06_clearedoffers.xls

/home/humed/python/fkdata/data/2006/2006_07_clearedoffers.xls

/home/humed/python/fkdata/data/2006/2006_08_clearedoffers.xls

/home/humed/python/fkdata/data/2006/2006_09_clearedoffers.xls
```

```
/home/humed/python/fkdata/data/2006/2006_10_clearedoffers.xls
/home/humed/python/fkdata/data/2006/2006_11_clearedoffers.xls
/home/humed/python/fkdata/data/2006/2006_12_clearedoffers.xls
/home/humed/python/fkdata/data/2007/2007_01_clearedoffers.xls
/home/humed/python/fkdata/data/2007/2007_02_clearedoffers.xls
/home/humed/python/fkdata/data/2007/2007_03_clearedoffers.xls
/home/humed/python/fkdata/data/2007/2007_04_clearedoffers.xls
/home/humed/python/fkdata/data/2007/2007_05_clearedoffers.xls
/home/humed/python/fkdata/data/2007/2007_06_clearedoffers.xls
/home/humed/python/fkdata/data/2007/2007_07_clearedoffers.xls
/home/humed/python/fkdata/data/2007/2007_08_clearedoffers.xls
/home/humed/python/fkdata/data/2007/2007_09_clearedoffers.xls
/home/humed/python/fkdata/data/2007/2007_10_clearedoffers.xls
/home/humed/python/fkdata/data/2007/2007_11_clearedoffers.xls
/home/humed/python/fkdata/data/2007/2007_12_clearedoffers.xls
/home/humed/python/fkdata/data/2008/2008_01_clearedoffers.xls
/home/humed/python/fkdata/data/2008/2008_02_clearedoffers.xls
/home/humed/python/fkdata/data/2008/2008_03_clearedoffers.xls
/home/humed/python/fkdata/data/2008/2008_04_clearedoffers.xls
/home/humed/python/fkdata/data/2008/2008_05_clearedoffers.xls
/home/humed/python/fkdata/data/2008/2008_06_clearedoffers.xls
/home/humed/python/fkdata/data/2008/2008_07_clearedoffers.xls
/home/humed/python/fkdata/data/2008/2008_08_clearedoffers.xls
/home/humed/python/fkdata/data/2008/2008_09_clearedoffers.xls
/home/humed/python/fkdata/data/2008/2008_10_clearedoffers.xls
/home/humed/python/fkdata/data/2008/2008_11_clearedoffers.xls
/home/humed/python/fkdata/data/2008/2008_12_clearedoffers.xls
/home/humed/python/fkdata/data/2009/2009_01_clearedoffers.xls
/home/humed/python/fkdata/data/2009/2009_02_clearedoffers.xls
/home/humed/python/fkdata/data/2009/2009_03_clearedoffers.xls
/home/humed/python/fkdata/data/2009/2009_04_clearedoffers.xls
/home/humed/python/fkdata/data/2009/2009_05_clearedoffers.xls
/home/humed/python/fkdata/data/2009/2009_06_clearedoffers.xls
/home/humed/python/fkdata/data/2009/2009_07_clearedoffers.xls
/home/humed/python/fkdata/data/2009/2009_08_clearedoffers.xls
/home/humed/python/fkdata/data/2009/2009_09_clearedoffers.xls
```

```
/home/humed/python/fkdata/data/2009/2009_10_clearedoffers.xls
/home/humed/python/fkdata/data/2009/2009_11_clearedoffers.xls
/home/humed/python/fkdata/data/2009/2009_12_clearedoffers.xls
/home/humed/python/fkdata/data/2010/2010_01_clearedoffers.xls
/home/humed/python/fkdata/data/2010/2010_02_clearedoffers.xls
/home/humed/python/fkdata/data/2010/2010_03_clearedoffers.xls
/home/humed/python/fkdata/data/2010/2010_04_clearedoffers.xls
/home/humed/python/fkdata/data/2010/2010_05_clearedoffers.xls
/home/humed/python/fkdata/data/2010/2010_06_clearedoffers.xls
/home/humed/python/fkdata/data/2010/2010_07_clearedoffers.xls
/home/humed/python/fkdata/data/2010/2010_08_clearedoffers.xls
/home/humed/python/fkdata/data/2010/2010_09_clearedoffers.xls
/home/humed/python/fkdata/data/2010/2010_10_clearedoffers.xls
/home/humed/python/fkdata/data/2010/2010_11_clearedoffers.xls
/home/humed/python/fkdata/data/2010/2010_12_clearedoffers.xls
/home/humed/python/fkdata/data/2011/2011_01_clearedoffers.xls
/home/humed/python/fkdata/data/2011/2011_02_clearedoffers.xls
/home/humed/python/fkdata/data/2011/2011_03_clearedoffers.xls
/home/humed/python/fkdata/data/2011/2011_04_clearedoffers.xls
/home/humed/python/fkdata/data/2011/2011_05_clearedoffers.xls
/home/humed/python/fkdata/data/2011/2011_06_clearedoffers.xls
/home/humed/python/fkdata/data/2011/2011_07_clearedoffers.xlsx
/home/humed/python/fkdata/data/2011/2011_08_clearedoffers.xlsx
/home/humed/python/fkdata/data/2011/2011_09_clearedoffers.xlsx
/home/humed/python/fkdata/data/2011/2011_10_clearedoffers.xlsx
/home/humed/python/fkdata/data/2011/2011_11_clearedoffers.xlsx
/home/humed/python/fkdata/data/2011/2011_12_clearedoffers.xlsx
/home/humed/python/fkdata/data/2012/2012_01_clearedoffers.xlsx
/home/humed/python/fkdata/data/2012/2012_02_clearedoffers.xlsx
/home/humed/python/fkdata/data/2012/2012_03_clearedoffers.xlsx
/home/humed/python/fkdata/data/2012/2012_04_clearedoffers.xlsx
/home/humed/python/fkdata/data/2012/2012_05_clearedoffers.xlsx
/home/humed/python/fkdata/data/2012/2012_06_clearedoffers.xlsx
/home/humed/python/fkdata/data/2012/2012_07_clearedoffers.xlsx
/home/humed/python/fkdata/data/2012/2012_08_clearedoffers.xlsx
CTCT!$A$1:$O$1278
```

```
True
/home/humed/python/fkdata/data/2012/2012_09_clearedoffers.xlsx

/home/humed/python/fkdata/data/2012/2012_10_clearedoffers.xlsx

/home/humed/python/fkdata/data/2012/2012_11_clearedoffers.xlsx

/home/humed/python/fkdata/data/2012/2012_12_clearedoffers.xlsx

/home/humed/python/fkdata/data/2013/2013_01_clearedoffers.xlsx
```

0.1.6 Parse the constrained on and off data

```
[NIcost,SIcost] = parse_data(fl_co,'constrainedcosts',block_island_map2)
NIcost['AMT'] = NIcost['AMT'].fillna(0) + NIcost['AMOUNT'].fillna(0)
SIcost['AMT'] = SIcost['AMT'].fillna(0) + SIcost['AMOUNT'].fillna(0)
NIcost = NIcost.drop(['BLOCK_ID','BLOCK_ID_x','BLOCK_ID_y','STATION','STATION_x','
    STATION_y','AMOUNT'], axis=1)
SIcost = SIcost.drop(['BLOCK_ID','BLOCK_ID_x','BLOCK_ID_y','STATION','STATION_x','
    STATION_y','AMOUNT'], axis=1)
```

```
/home/humed/python/fkdata/data/2005/2005_09_constrainedcosts.xls
/home/humed/python/fkdata/data/2005/2005_10_constrainedcosts.xls

/home/humed/python/fkdata/data/2005/2005_11_constrainedcosts.xls

/home/humed/python/fkdata/data/2005/2005_12_constrainedcosts.xls

/home/humed/python/fkdata/data/2006/2006_01_constrainedcosts.xls

/home/humed/python/fkdata/data/2006/2006_02_constrainedcosts.xls

/home/humed/python/fkdata/data/2006/2006_03_constrainedcosts.xls

/home/humed/python/fkdata/data/2006/2006_04_constrainedcosts.xls

/home/humed/python/fkdata/data/2006/2006_05_constrainedcosts.xls

/home/humed/python/fkdata/data/2006/2006_06_constrainedcosts.xls

/home/humed/python/fkdata/data/2006/2006_07_constrainedcosts.xls

/home/humed/python/fkdata/data/2006/2006_08_constrainedcosts.xls

/home/humed/python/fkdata/data/2006/2006_09_constrainedcosts.xls

/home/humed/python/fkdata/data/2006/2006_10_constrainedcosts.xls

/home/humed/python/fkdata/data/2006/2006_11_constrainedcosts.xls

/home/humed/python/fkdata/data/2006/2006_12_constrainedcosts.xls

/home/humed/python/fkdata/data/2007/2007_01_constrainedcosts.xls

/home/humed/python/fkdata/data/2007/2007_02_constrainedcosts.xls

/home/humed/python/fkdata/data/2007/2007_03_constrainedcosts.xls

/home/humed/python/fkdata/data/2007/2007_04_constrainedcosts.xls

/home/humed/python/fkdata/data/2007/2007_05_constrainedcosts.xls

/home/humed/python/fkdata/data/2007/2007_06_constrainedcosts.xls
```

```
/home/humed/python/fkdata/data/2007/2007_07_constrainedcosts.xls
/home/humed/python/fkdata/data/2007/2007_08_constrainedcosts.xls
/home/humed/python/fkdata/data/2007/2007_09_constrainedcosts.xls
/home/humed/python/fkdata/data/2007/2007_10_constrainedcosts.xls
/home/humed/python/fkdata/data/2007/2007_11_constrainedcosts.xls
/home/humed/python/fkdata/data/2007/2007_12_constrainedcosts.xls
/home/humed/python/fkdata/data/2008/2008_01_constrainedcosts.xls
/home/humed/python/fkdata/data/2008/2008_02_constrainedcosts.xls
/home/humed/python/fkdata/data/2008/2008_03_constrainedcosts.xls
/home/humed/python/fkdata/data/2008/2008_04_constrainedcosts.xls
/home/humed/python/fkdata/data/2008/2008_05_constrainedcosts.xls
/home/humed/python/fkdata/data/2008/2008_06_constrainedcosts.xls
/home/humed/python/fkdata/data/2008/2008_07_constrainedcosts.xls
/home/humed/python/fkdata/data/2008/2008_08_constrainedcosts.xls
/home/humed/python/fkdata/data/2008/2008_09_constrainedcosts.xls
/home/humed/python/fkdata/data/2008/2008_10_constrainedcosts.xls
/home/humed/python/fkdata/data/2008/2008_11_constrainedcosts.xls
/home/humed/python/fkdata/data/2008/2008_12_constrainedcosts.xls
/home/humed/python/fkdata/data/2009/2009_01_constrainedcosts.xls
/home/humed/python/fkdata/data/2009/2009_02_constrainedcosts.xls
/home/humed/python/fkdata/data/2009/2009_03_constrainedcosts.xls
/home/humed/python/fkdata/data/2009/2009_04_constrainedcosts.xls
/home/humed/python/fkdata/data/2009/2009_05_constrainedcosts.xls
/home/humed/python/fkdata/data/2009/2009_06_constrainedcosts.xls
/home/humed/python/fkdata/data/2009/2009_07_constrainedcosts.xls
/home/humed/python/fkdata/data/2009/2009_08_constrainedcosts.xls
/home/humed/python/fkdata/data/2009/2009_09_constrainedcosts.xls
/home/humed/python/fkdata/data/2009/2009_10_constrainedcosts.xls
/home/humed/python/fkdata/data/2009/2009_11_constrainedcosts.xls
/home/humed/python/fkdata/data/2009/2009_12_constrainedcosts.xls
/home/humed/python/fkdata/data/2010/2010_01_constrainedcosts.xls
/home/humed/python/fkdata/data/2010/2010_02_constrainedcosts.xls
/home/humed/python/fkdata/data/2010/2010_03_constrainedcosts.xls
/home/humed/python/fkdata/data/2010/2010_04_constrainedcosts.xls
/home/humed/python/fkdata/data/2010/2010_05_constrainedcosts.xls
/home/humed/python/fkdata/data/2010/2010_06_constrainedcosts.xls
```

```
/home/humed/python/fkdata/data/2010/2010_07_constrainedcosts.xls
/home/humed/python/fkdata/data/2010/2010_08_constrainedcosts.xls
/home/humed/python/fkdata/data/2010/2010_09_constrainedcosts.xls
/home/humed/python/fkdata/data/2010/2010_10_constrainedcosts.xls
/home/humed/python/fkdata/data/2010/2010_11_constrainedcosts.xls
/home/humed/python/fkdata/data/2010/2010_12_constrainedcosts.xls
/home/humed/python/fkdata/data/2011/2011_01_constrainedcosts.xls
/home/humed/python/fkdata/data/2011/2011_02_constrainedcosts.xls
/home/humed/python/fkdata/data/2011/2011_03_constrainedcosts.xls
/home/humed/python/fkdata/data/2011/2011_04_constrainedcosts.xls
/home/humed/python/fkdata/data/2011/2011_05_constrainedcosts.xls
/home/humed/python/fkdata/data/2011/2011_06_constrainedcosts.xls
/home/humed/python/fkdata/data/2011/2011_07_constrainedcosts.xls
/home/humed/python/fkdata/data/2011/2011_08_constrainedcosts.xls
/home/humed/python/fkdata/data/2011/2011_09_constrainedcosts.xls
/home/humed/python/fkdata/data/2011/2011_10_constrainedcosts.xls
/home/humed/python/fkdata/data/2011/2011_11_constrainedcosts.xls
/home/humed/python/fkdata/data/2011/2011_12_constrainedcosts.xls
/home/humed/python/fkdata/data/2012/2012_01_constrainedcosts.xls
/home/humed/python/fkdata/data/2012/2012_02_constrainedcosts.xls
/home/humed/python/fkdata/data/2012/2012_03_constrainedcosts.xls
/home/humed/python/fkdata/data/2012/2012_04_constrainedcosts.xls
/home/humed/python/fkdata/data/2012/2012_05_constrainedcosts.xls
/home/humed/python/fkdata/data/2012/2012_06_constrainedcosts.xls
/home/humed/python/fkdata/data/2012/2012_07_constrainedcosts.xls
/home/humed/python/fkdata/data/2012/2012_08_constrainedcosts.xls
/home/humed/python/fkdata/data/2012/2012_09_constrainedcosts.xls
/home/humed/python/fkdata/data/2012/2012_10_constrainedcosts.xls
/home/humed/python/fkdata/data/2012/2012_11_constrainedcosts.xls
/home/humed/python/fkdata/data/2012/2012_12_constrainedcosts.xls
/home/humed/python/fkdata/data/2013/2013_01_constrainedcosts.xls
```

0.1.7 Data integrity check – constrained costs...

```
#eyeball
NIcost.head(10)
```

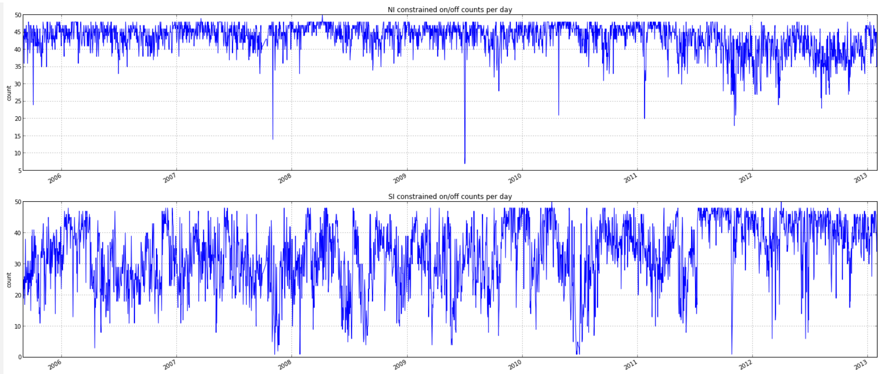
```
                                    AMT CON_IND GIP_GXP_FULL ORG_CODE  PRICE  QUANTITY
TRADING_DATE TRADING_PERIOD
2005-09-01   1               -252.0435      OF          WTO     MRPL -10.08    50.000
             2               -249.7637      OF          WTO     MRPL  -9.99    50.000
             3               -303.8478      OF          WTO     MRPL -12.69    47.880
             4                163.7578      ON          HLY     GENE  20.57    15.922
             5                 13.8348      ON          WTO     MRPL   3.24     8.540
             6                 26.0295      ON          WTO     MRPL   3.35    15.540
             7                 40.5345      ON          WTO     MRPL   3.38    24.020
             8                  8.8244      ON          WTO     MRPL   0.83    21.140
             9                527.0195      ON          HLY     GENE  21.08    50.000
             10                82.3481      ON          HLY     GENE  20.08     8.202
```

```
SIcost.tail(10)
```

```
                                    AMT CON_IND GIP_GXP_FULL ORG_CODE  PRICE  QUANTITY
TRADING_DATE TRADING_PERIOD
2013-01-31   37              -796.6639      OF          WTR     MERI -69.42    22.952
             38              -780.5000      OF          WTR     MERI -62.44    25.000
             39              -564.0869      OF          WTR     MERI -55.90    20.182
             40              -654.1250      OF          WTR     MERI -52.33    25.000
             41                 0.0136      ON          CLU     CTCT   0.16     0.170
             42               -42.8750      OF          CLU     CTCT  -3.43    25.000
             44               105.6950      ON          WTR     MERI  63.98     3.304
             45              -257.4452      OF          CLU     CTCT -20.60    25.000
             46               -18.5133      OF          CLU     CTCT  -2.82    13.130
             47               -14.5452      OF          CLU     CTCT  -2.83    10.270
```

```
#ok, looks like there is missing data, lets count periods per day
fig=plt.figure(1,figsize=[27,12])
ax=fig.add_subplot(211)
NIcost2=NIcost.groupby(level=[0,1]).sum()
NIcost2.groupby(NIcost2.index.map(lambda x: x[0])).count().AMT.plot(ax=ax)
title('NI constrained on/off counts per day')
ylabel('count')
ax2=fig.add_subplot(212)
SIcost2=SIcost.groupby(level=[0,1]).sum()
SIcost2.groupby(SIcost2.index.map(lambda x: x[0])).count().AMT.plot(ax=ax2)
title('SI constrained on/off counts per day')
ylabel('count')
```

```
<matplotlib.text.Text at 0x7adf690>
```



So either we have missing data for constrained on/off costs, or, there are periods when there are no constrained on/off costs?

This is much more likely, see http://www.systemoperator.co.nz/f1693,2039326/frequency-keeping-costs-17-apr-08.pdf

Lets check if there are any constrained on/off costs at zero.

```
SIcost2[SIcost2.AMT==0].head()
```

```
                            AMT  PRICE  QUANTITY
TRADING_DATE TRADING_PERIOD
2011-03-26   1                0      0        25
             2                0      0        25
             3                0      0        25
             4                0      0        25
             5                0      0        25
```

So constrained on/off costs are only reported when they are non-zero (with the exception of 26 March, 2011)

We will reindex this cost data and then fill all nans with 0, but first lets eyeball the offers data

```
#Eyeball
NI.head(10)
```

```
                           trader block or market node  frequency qty partperiod  \
trading date trading period
2005-09-01   1                MRPL                 WTO            50        NaN
             2                MRPL                 WTO            50        NaN
             3                MRPL                 WTO            50        NaN
             4                GENE        HLY2201 HLY1            16          C
             4                GENE        HLY2201 HLY3            17          C
             4                GENE        HLY2201 HLY4            17          C
             5                MRPL                 WTO            50          C
             6                MRPL                 WTO            50        NaN
             7                MRPL                 WTO            50        NaN
             8                MRPL                 WTO            50        NaN


                             offer
```

```
trading date trading period
2005-09-01   1                 1490
             2                 1490
             3                 1490
             4                  400
             4                  400
             4                  400
             5                 1190
             6                 1190
             7                 1190
             8                 1190
```

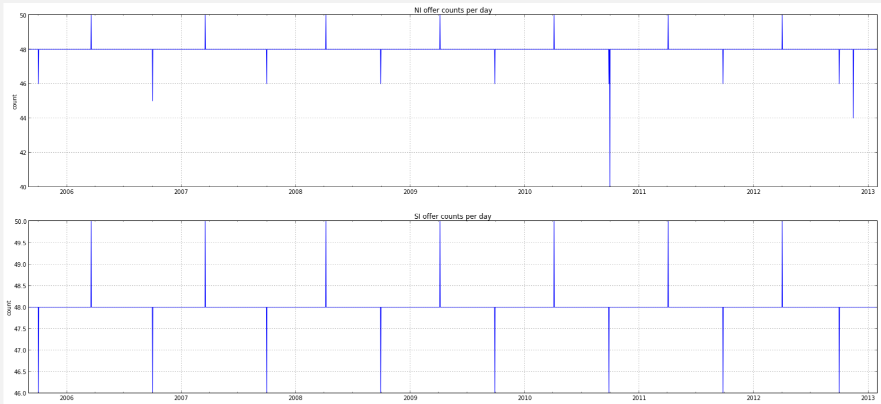
```
SI.tail(10)
```

```
                              trader block or market node  frequency qty partperiod  \
trading date trading period
2013-01-31   39                 MERI                   WTR             25       None
             40                 MERI                   WTR             25       None
             41                 CTCT                   CLU             25          C
             42                 CTCT                   CLU             25       None
             43                 CTCT                   CLU             25       None
             44                 MERI                   WTR             25          C
             45                 CTCT                   CLU             25          C
             46                 CTCT                   CLU             25       None
             47                 CTCT                   CLU             25       None
             48                 CTCT                   CLU             25       None

                               offer
trading date trading period
2013-01-31   39                  200
             40                  250
             41                  299
             42                  299
             43                  299
             44                 5000
             45                  299
             46                  299
             47                  299
             48                  299
```

```
#This looks better than constrained data, lets do a daily count anyways...
fig=plt.figure(2,figsize=[27,12])
ax=fig.add_subplot(211)
NIoff=NI.groupby(level=[0,1]).sum()
NIoff.groupby(NIoff.index.map(lambda x: x[0])).count().offer.plot(ax=ax)
title('NI offer counts per day')
ylabel('count')

ax2=fig.add_subplot(212)
SIoff=SI.groupby(level=[0,1]).sum()
SIoff.groupby(SIoff.index.map(lambda x: x[0])).count().offer.plot(ax=ax2)
```

```
title('SI offer counts per day')
ylabel('count')
```

```
<matplotlib.text.Text at 0xe002410>
```



Data integrity check for offers... South Island offer data appears complete over ther entire duration (amazing)...

North Island offer data has the odd trading period missing: missing row (at 2am) on the short day, 1 Oct, 2006 8 missing data rows on 29th Sep, 2010 4 missing data rows on 14th Nov, 2012 Note: the NIoff and SIoff dataframes are summed over all duplicate index values.

We do this to get a total offer per trading period over all cleared offers, (yes there can be multiple units dispatched to make up the 50MW/25MW FK band required).

For example, look at some recent NI offer data, check out TP 39,40 and 44 for the two examples below.

```
NI.tail(14)
```

```
                             trader block or market node  frequency qty partperiod  \
trading date trading period
2013-01-31   38                MRPL                   WTO            50       None
             39                CTCT         SFD2201 SFD21            25          C
             39                CTCT         SFD2201 SFD22            25          C
             40                CTCT         SFD2201 SFD21            25       None
             40                CTCT         SFD2201 SFD22            25       None
             41                MRPL                   WTO            50          C
             42                MRPL                   WTO            50       None
             43                GENE                   WKA            50          C
             44                CTCT         SFD2201 SFD21            33          C
             44                CTCT         SFD2201 SFD22            20          C
             45                MRPL                   WTO            50          C
             46                GENE          TKU2201 TKU0            50          C
             47                GENE          TKU2201 TKU0            50       None
             48                GENE          TKU2201 TKU0            50       None

                                offer
trading date trading period
2013-01-31   38              1800.00
```

```
             39                 950.00
             39                 950.00
             40                 950.00
             40                 950.00
             41                1800.00
             42                1800.00
             43                3000.06
             44                1150.00
             44                 770.00
             45                1700.00
             46                1250.06
             47                1250.06
             48                1250.06
```

```
NIoff.tail(11)
```

```
                             frequency qty    offer
trading date trading period
2013-01-31   38                         50  1800.00
             39                         50  1900.00
             40                         50  1900.00
             41                         50  1800.00
             42                         50  1800.00
             43                         50  3000.06
             44                         53  1920.00
             45                         50  1700.00
             46                         50  1250.06
             47                         50  1250.06
             48                         50  1250.06
```

We have lost data by trying to sum string columns in NIoff. We can do better than this. Lets add the first trader when there are duplicates.

```
NIoff['trader'] = NI.groupby(level=[0,1]).first().trader
SIoff['trader'] = SI.groupby(level=[0,1]).first().trader
```

```
NIoff.tail(14)
```

```
                             frequency qty    offer trader
trading date trading period
2013-01-31   35                         50  1650.00   MRPL
             36                         50  1700.00   MRPL
             37                         50  1750.00   MRPL
             38                         50  1800.00   MRPL
             39                         50  1900.00   CTCT
             40                         50  1900.00   CTCT
             41                         50  1800.00   MRPL
             42                         50  1800.00   MRPL
             43                         50  3000.06   GENE
             44                         53  1920.00   CTCT
             45                         50  1700.00   MRPL
```

```
            46                          50  1250.06   GENE
            47                          50  1250.06   GENE
            48                          50  1250.06   GENE
```

```
#Ok, this looks good, but lets just make sure that there are always only one trader when
     we have duplicates.
NIoff['trader2'] = NI.groupby(level=[0,1]).last().trader
```

```
NIoff[NIoff.trader!=NIoff.trader2]
```

```
Empty DataFrame
Columns: [frequency qty, offer, trader, trader2]
Index: []
```

0.1.8 Reindex data

As we have a complete dataset for the SI offers we will reindex all other (summed duplicate) data to this index.

```
NIoff=NIoff.reindex(index=SIoff.index)
SIcost2 = SIcost2.reindex(index=SIoff.index).copy()
NIcost2 = NIcost2.reindex(index=SIoff.index).copy()
```

0.1.9 Time series conversion (for plotting)

One of the problems we have is that time series plots don't tend to plot well if we use a multi-index based on Date and Trading period. What we need is a function that takes a multi-index date/trading period time stamped data and returns a time series. Sounds easy, but if we are to do it properly its a real headache because of day-light savings time.

In gnash.py I've had a first crack at a function to do this the gnash way.

Called timeseries_convert, input is a multiindex df with (date,TP) index, converts to a datetime index with the same times as used by Gnash on the long and short days. Woo hoo.

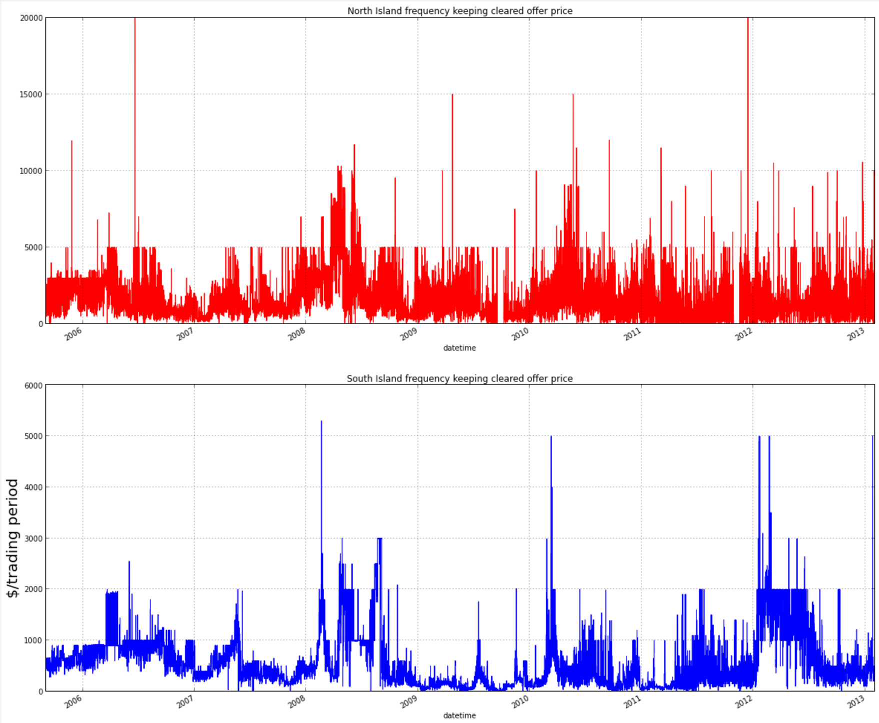
```
SIoff_ts=timeseries_convert(SIoff) #get our reindexed time series data from the
    multiindexed data
NIoff_ts=timeseries_convert(NIoff)
```

0.2 Offer plots

```
#Time series - all data
fig=plt.figure(3,figsize=[20,18])
ax1=fig.add_subplot(211)
NIoff_ts.offer.plot(style='r',ax=ax1)
#ylim([0,6000])
title('North Island frequency keeping cleared offer price')
ax2=fig.add_subplot(212,sharex=ax1)
SIoff_ts.offer.plot(style='b',ax=ax2)
```

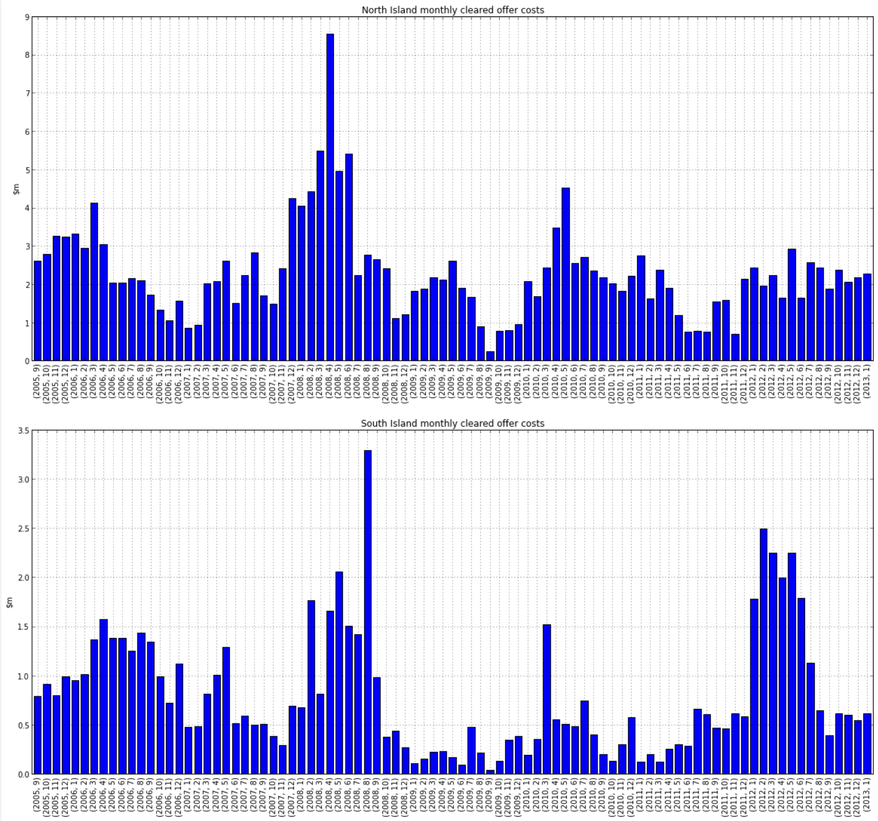
```
#ylim([0,6000])
ylabel('$/trading period',fontsize=20)
title('South Island frequency keeping cleared offer price')
```

```
<matplotlib.text.Text at 0x584e350>
```



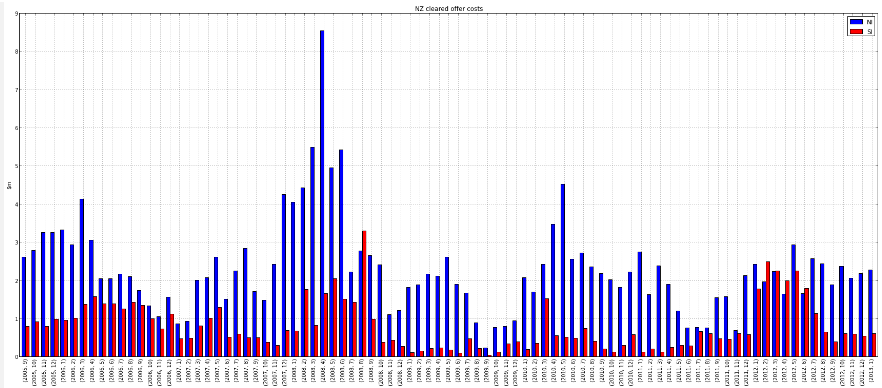

```
#Monthly bar plot
fig=plt.figure(4,figsize=[20,18])
ax=fig.add_subplot(211)
(NIoff.offer.groupby(NIoff.index.map(lambda x: (x[0].year,x[0].month))).sum()/1000000.0)
    .plot(kind='bar',ax=ax)
ylabel('$m')
title('North Island monthly cleared offer costs')
ax2=fig.add_subplot(212)
(SIoff.offer.groupby(SIoff.index.map(lambda x: (x[0].year,x[0].month))).sum()/1000000.0)
    .plot(kind='bar',ax=ax2)
ylabel('$m')
title('South Island monthly cleared offer costs')
```

```
<matplotlib.text.Text at 0x777ad50>
```



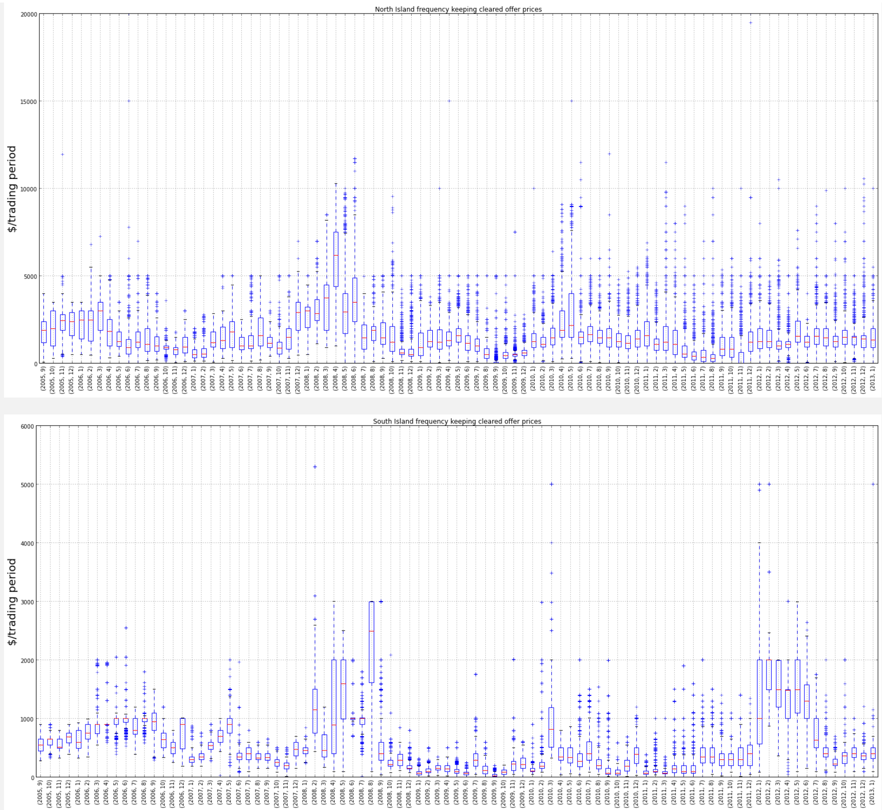

```
#Monthly bar plot with dual bars
monthlys=(DataFrame({'NI':NI.offer.groupby(NI.index.map(lambda x: (x[0].year,x[0].month)
    )).sum(),'SI':SI.offer.groupby(SI.index.map(lambda x: (x[0].year,x[0].month))).sum()
    })/1000000.0)
fig=plt.figure(5,figsize=[30,12])
ax=fig.add_subplot(111)
monthlys.plot(kind='bar',ax=ax)
title('NZ cleared offer costs')
legend()
grid('on')
ylabel('$m')
```

```
<matplotlib.text.Text at 0x7bdda10>
```




```
#monthly boxplots
#North
fig=plt.figure(6,figsize=[27,12])
ax1=fig.add_subplot(211)
NIoff.boxplot(column='offer',by=lambda x: (x[0].year,x[0].month),ax=ax1)
title('North Island frequency keeping cleared offer prices')
xlabel('')
ylabel('$/trading period',fontsize=20)
#title('')
suptitle('')
plt.tick_params(axis='x', which='major', labelsize=10)
plt.xticks(rotation=90 )
#South
fig=plt.figure(7,figsize=[27,12])
ax2=fig.add_subplot(212)
SIoff.boxplot(column='offer',by=lambda x: (x[0].year,x[0].month),ax=ax2)
title('South Island frequency keeping cleared offer prices')
xlabel('')
ylabel('$/trading period',fontsize=20)
#title('')
suptitle('')
plt.tick_params(axis='x', which='major', labelsize=10)
plt.xticks(rotation=90 )
```

```
(array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
       35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51,
       52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68,
       69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85,
       86, 87, 88, 89]),
 <a list of 89 Text xticklabel objects>)
```

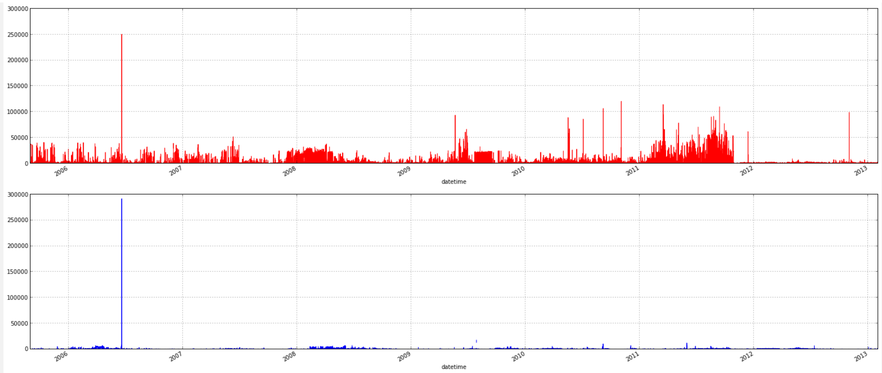


0.3 Constrained on/off costs

```
#get nice time series data
SIcost2_ts=timeseries_convert(SIcost2)
NIcost2_ts=timeseries_convert(NIcost2)
```

```
#plot total constrained, both on and off together
fig=plt.figure(8,figsize=[27,12])
ax1=fig.add_subplot(211)
abs(NIcost2_ts.AMT).plot(style='r',ax=ax1)
ax2=fig.add_subplot(212,sharex=ax1)
abs(SIcost2_ts.AMT).plot(style='b',ax=ax2)
```

```
<matplotlib.axes.AxesSubplot at 0x18148d90>
```



0.4 totals

```
NIoff
```

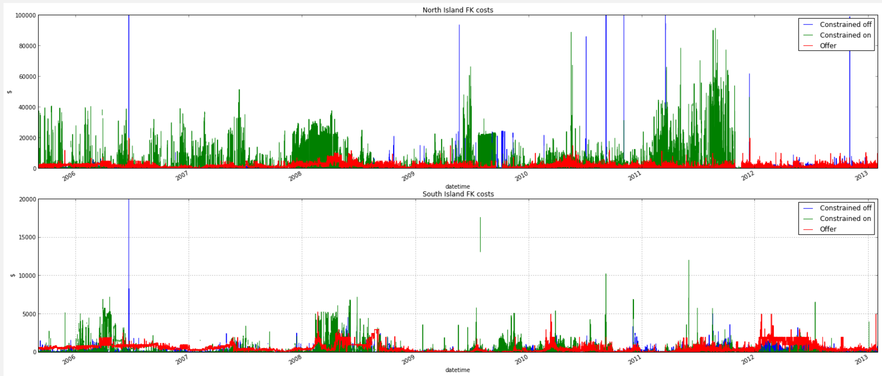
```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 130078 entries, (2005-09-01 00:00:00, 1.0) to (2013-01-31 00:00:00, 48.0)
Data columns:
frequency qty    130065  non-null values
offer            130065  non-null values
trader           130065  non-null values
trader2          130065  non-null values
datetime         130078  non-null values
dtypes: datetime64[ns](1), float64(2), object(2)
```

```
#create dataframe with all FK costs
total_ni=DataFrame({'Constrained off':abs(NIcost2.AMT[NIcost2.AMT<0]),'Constrained on':
    abs(NIcost2.AMT[NIcost2.AMT>0]),'Offer':NIoff.offer,'Trader':NIoff.trader})
total_si=DataFrame({'Constrained off':abs(SIcost2.AMT[SIcost2.AMT<0]),'Constrained on':
    abs(SIcost2.AMT[SIcost2.AMT>0]),'Offer':SIoff.offer,'Trader':SIoff.trader})
#time series
total_ni_ts=DataFrame({'Constrained off':abs(NIcost2_ts.AMT[NIcost2_ts.AMT<0]),'
    Constrained on':abs(NIcost2_ts.AMT[NIcost2_ts.AMT>0]),'Offer':NIoff_ts.offer})
total_si_ts=DataFrame({'Constrained off':abs(SIcost2_ts.AMT[SIcost2_ts.AMT<0]),'
    Constrained on':abs(SIcost2_ts.AMT[SIcost2_ts.AMT>0]),'Offer':SIoff_ts.offer})
```

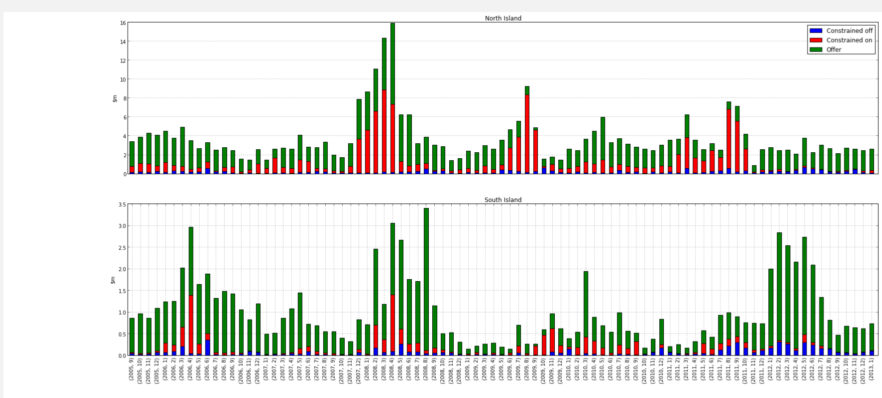
```
#time series plots
fig=plt.figure(9,figsize=[27,12])
ax1=fig.add_subplot(211)
ylim([0,100000])
grid('on')
title('North Island FK costs')
ylabel('$')
total_ni_ts.plot(ax=ax1)
ax2=fig.add_subplot(212,sharex=ax1)
total_si_ts.plot(ax=ax2)
ylim([0,20000])
grid('on')
```

```
title('South Island FK costs')
ylabel('$')
```

```
<matplotlib.text.Text at 0x18366250>
```



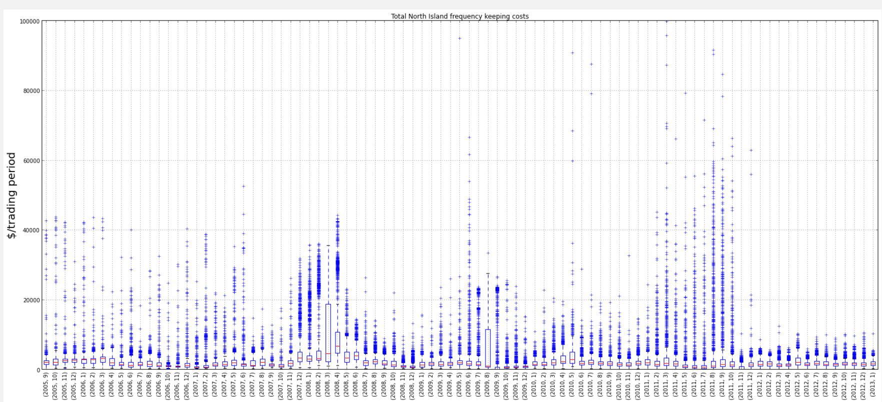

```
#monthly stacked bar plot
fig=plt.figure(10,figsize=[27,12])
ax1=fig.add_subplot(211)
(total_ni_ts.groupby(total_ni_ts.index.map(lambda x: (x.year,x.month))).sum()/1000000.0)
    .plot(kind='bar',stacked=True,ax=ax1)
grid('on')
ax1.axes.get_xaxis().set_visible(False)
ylabel('$m')
title('North Island')
ax2=fig.add_subplot(212)
(total_si_ts.groupby(total_si_ts.index.map(lambda x: (x.year,x.month))).sum()/1000000.0)
    .plot(kind='bar',stacked=True,ax=ax2)
grid('on')
ylabel('$m')
title('South Island')
ax2.legend().set_visible(False)
```

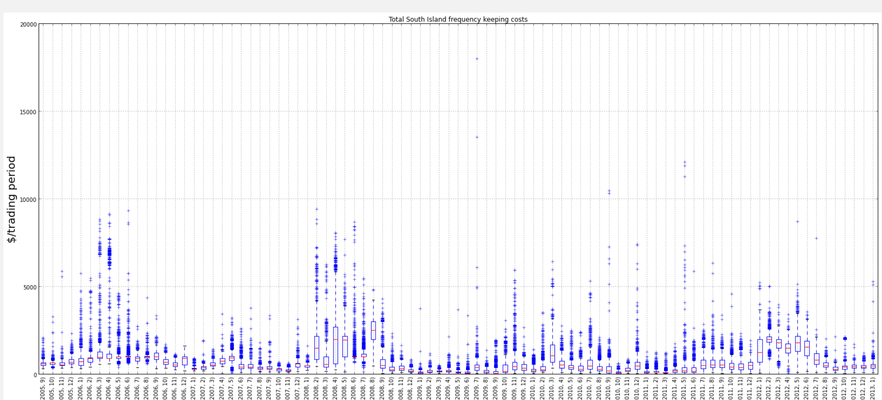



```
#monthly boxplots
fig=plt.figure(11,figsize=[27,12])
```

```
ax1=fig.add_subplot(111)
total_ni['total']=total_ni.sum(axis=1)
total_ni.boxplot(column='total',by=lambda x: (x[0].year,x[0].month),ax=ax1)
title('Total North Island frequency keeping costs')
xlabel('')
ylabel('$/trading period',fontsize=20)
suptitle('')
plt.tick_params(axis='x', which='major', labelsize=10)
plt.xticks(rotation=90 )
ylim([0,100000])

#South
total_si['total']=total_si.sum(axis=1)
fig=plt.figure(12,figsize=[27,12])
ax2=fig.add_subplot(111)
total_si.boxplot(column='total',by=lambda x: (x[0].year,x[0].month),ax=ax2)
title('Total South Island frequency keeping costs')
xlabel('')
ylabel('$/trading period',fontsize=20)
#title('')
suptitle('')
plt.tick_params(axis='x', which='major', labelsize=10)
plt.xticks(rotation=90 )
ylim([0,20000])
```

```
(0, 20000)
```





0.5 Annual costs

Frequency keeping in NZ is a lucrative market, espcially in the North Island. Total annual prices are:

```
#North Island ($ millions)
annual_ni = total_ni_ts.groupby(total_ni_ts.index.map(lambda x: x.year)).sum()/1000000.0
annual_si = total_si_ts.groupby(total_si_ts.index.map(lambda x: x.year)).sum()/1000000.0
```

```
annual_ni['total']=annual_ni.sum(axis=1)
annual_ni
```

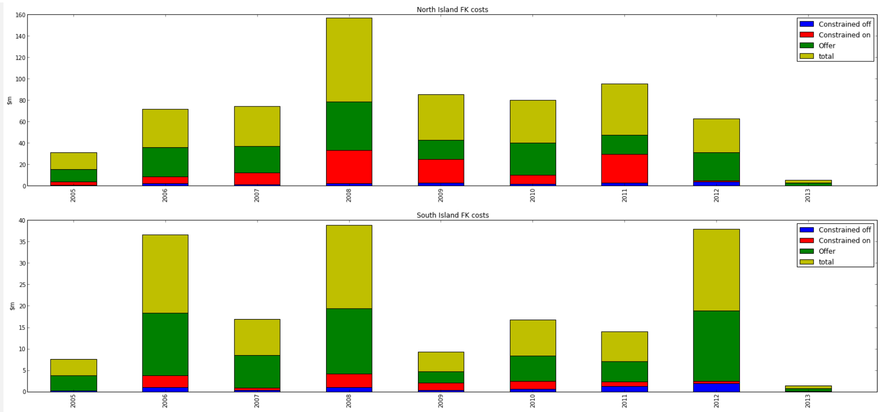
	Constrained off	Constrained on	Offer	total
2005	0.736679	2.947449	11.920348	15.604475
2006	2.294414	6.103282	27.481446	35.879141
2007	1.260190	10.937436	24.959288	37.156914
2008	2.282977	30.818306	45.264662	78.365945
2009	2.719450	22.250095	17.822069	42.791614
2010	1.643899	8.480448	30.065261	40.189607
2011	2.892774	26.574512	18.107925	47.575211
2012	3.719632	1.282409	26.365590	31.367632
2013	0.087403	0.252601	2.279261	2.619265

```
annual_si['total']=annual_si.sum(axis=1)
annual_si
```

	Constrained off	Constrained on	Offer	total
2005	0.166960	0.100713	3.501293	3.768966
2006	1.020732	2.727804	14.557619	18.306155
2007	0.370023	0.526671	7.572283	8.468976
2008	0.982658	3.183087	15.260312	19.426057
2009	0.297761	1.750516	2.605794	4.654071
2010	0.562797	1.853484	5.980894	8.397175
2011	1.224271	1.072421	4.700201	6.996892
2012	1.923130	0.523942	16.490445	18.937517
2013	0.099248	0.017012	0.612341	0.728600

```
fig=plt.figure(13,figsize=[27,12])
ax1=fig.add_subplot(211)
annual_ni.plot(kind='bar',stacked=True,ax=ax1)
title('North Island FK costs')
ylabel('$m')
ax2=fig.add_subplot(212)
annual_si.plot(kind='bar',stacked=True,ax=ax2)
title('South Island FK costs')
ylabel('$m')
```

```
<matplotlib.text.Text at 0x15548190>
```



North Island price's tend to be much higher than SI prices in the two FK markets.

```
(annual_ni.total/annual_si.total).describe()
```

```
count    9.000000
mean     4.505889
std      2.321728
min      1.656375
25%      3.594928
50%      4.140253
75%      4.786087
max      9.194448
Dtype: float64
```

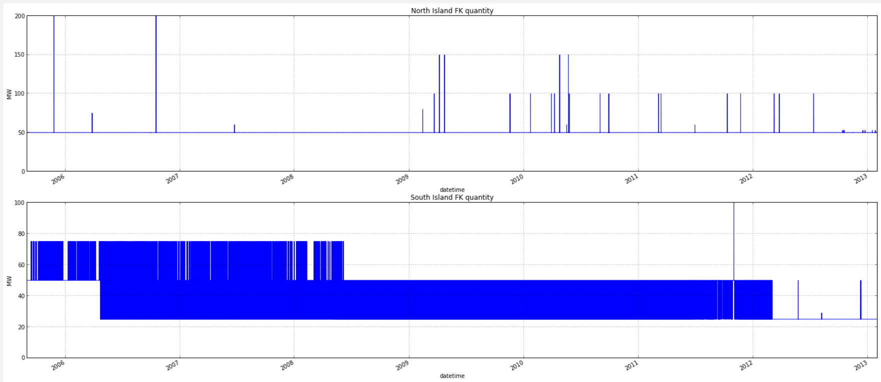
So, on average, NI prices have been historically over 4 times the SI prices in 2009 were 9 times the NI price. High SI FK offer prices in 2012 brought this ratio down to just 1.65.

0.5.1 Normalize costs with FK band quantity

If we normalize this based on the required FK band this will get even worse. ALthough the NI has, for the duration of this dataset, had and pretty constant FK band of 50MW, the SI has had decreased the FK band required from 75MW during certain hours in 2005/2006 to between 50MW and 25MW between 2008 and 2012 to a fixed 25MW in early 2012. To get a better comparison of FK costs between the two markets we should probably normalise the costs on the FK band.

```
fig=plt.figure(14,figsize=[27,12])
ax1=fig.add_subplot(211)
NIoff_ts['frequency qty'].plot(ax=ax1)
title('North Island FK quantity')
ylabel('MW')
ylim([0,200])
ax2=fig.add_subplot(212)
SIoff_ts['frequency qty'].plot(ax=ax2)
title('South Island FK quantity')
ylabel('MW')
ylim([0,100])
```

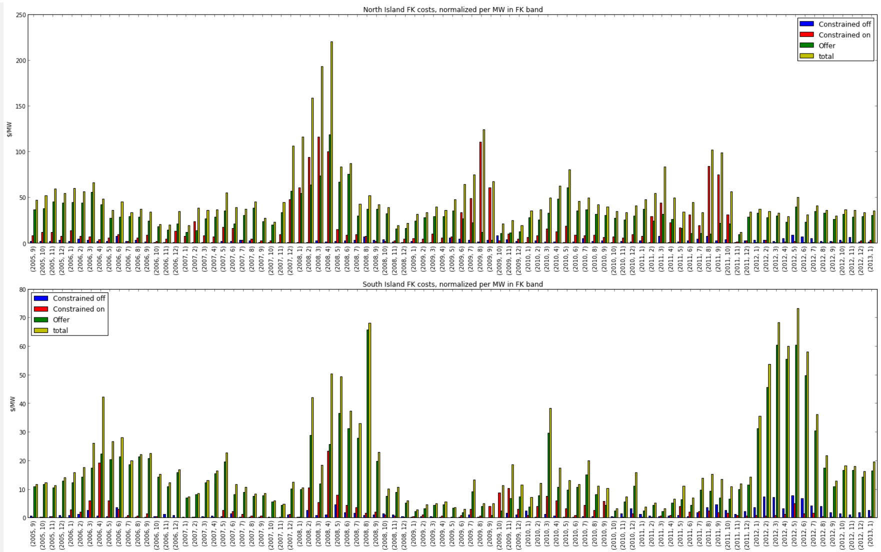
```
(0, 100)
```




```
#add quantity
total_si['quantity']=SIoff['frequency qty']
total_ni['quantity']=NIoff['frequency qty']
#normalise on quantity
total_si_norm=DataFrame({'Offer':total_si.Offer/total_si.quantity,'Constrained on':
    total_si['Constrained on']/total_si.quantity,'Constrained off':total_si['Constrained
    off']/total_si.quantity})
total_ni_norm=DataFrame({'Offer':total_ni.Offer/total_ni.quantity,'Constrained on':
    total_ni['Constrained on']/total_ni.quantity,'Constrained off':total_ni['Constrained
    off']/total_ni.quantity})
total_si_norm=total_si_norm.fillna(0)
total_ni_norm=total_ni_norm.fillna(0)
total_si_norm['total'] = total_si_norm.sum(axis=1)
total_ni_norm['total'] = total_ni_norm.sum(axis=1)
```

```
fig=plt.figure(15,figsize=[27,16])
ax1=fig.add_subplot(211)
total_ni_norm.groupby(total_ni_norm.index.map(lambda x: (x[0].year,x[0].month))).mean().
    plot(kind='bar',ax=ax1)
title('North Island FK costs, normalized per MW in FK band')
ylabel('$/MW')
ax2=fig.add_subplot(212)
total_si_norm.groupby(total_si_norm.index.map(lambda x: (x[0].year,x[0].month))).mean().
    plot(kind='bar',ax=ax2)
title('South Island FK costs, normalized per MW in FK band')
ylabel('$/MW')
```

```
<matplotlib.text.Text at 0x24e2b550>
```




```
#lets redo the ratios above
annual_ni_norm = total_ni_norm.groupby(total_ni_norm.index.map(lambda x: x[0].year)).
    mean()
annual_si_norm = total_si_norm.groupby(total_si_norm.index.map(lambda x: x[0].year)).
    mean()
```

```
annual_ni_norm
```

```
      Constrained off  Constrained on      Offer      total
2005         2.516774       10.069862  40.694834  53.281471
2006         2.619190        6.961682  31.365673  40.946544
2007         1.438573       12.476970  28.491809  42.407353
2008         2.599017       35.084593  51.530808  89.214418
2009         3.103893       25.398234  20.327964  48.830091
2010         1.875815        9.679862  34.286295  45.841971
2011         3.301222       30.335403  20.655242  54.291867
2012         4.231667        1.459935  30.001192  35.692794
2013         1.173676        3.392662  30.550496  35.116834
```

```
annual_si_norm
```

```
      Constrained off  Constrained on      Offer      total
2005         0.554283        0.335395  11.516981  12.406660
2006         1.065319        3.551871  17.516282  22.133472
2007         0.468483        0.850999   9.604965  10.924447
2008         1.487870        5.077980  23.264586  29.830436
2009         0.496262        2.779572   4.204385   7.480219
2010         0.932573        3.082053   9.960921  13.975547
2011         1.688027        1.459825   6.597053   9.744905
2012         4.171618        1.109684  34.022145  39.303447
2013         2.667944        0.457304  16.460780  19.586028
```

```
(annual_ni_norm.total/annual_si_norm.total)
```

```
2005    4.294586
2006    1.849983
2007    3.881876
2008    2.990718
2009    6.527896
2010    3.280156
2011    5.571308
2012    0.908134
2013    1.792953
Name: total, Dtype: float64
```

```
(annual_ni_norm.total/annual_si_norm.total).describe()
```

```
count    9.000000
mean     3.455290
std      1.833020
min      0.908134
25%      1.849983
50%      3.280156
75%      4.294586
max      6.527896
Dtype: float64
```

So, on ave. NI FK over 3 times the cost of SI FK.

In 2012 FK costs (per MW) were in fact higher in the SI than in the NI.

Constrained on payments were gamed in 2008/2009 and 2011.

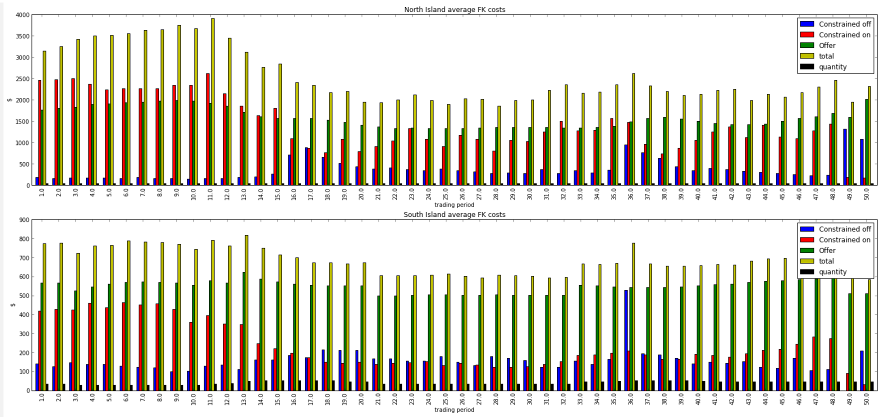
This was achieved by constructing generator offer price curves which increased after the mid-point of the frequency keeping band and whose costs were hidden from the selection algorithm. This was fixed in 2011 by adjusting the selection algorithm to included offers in the whole fk band.

0.5.2 Trading period variability

There are a few other aspects we haven't looked at: trading period variability and trader variability.

```
fig=plt.figure(16,figsize=[27,12])
ax1=fig.add_subplot(211)
total_ni.groupby(level=1).mean().plot(kind='bar',ax=ax1)
title('North Island average FK costs')
ylabel('$')
ax2=fig.add_subplot(212)
total_si.groupby(level=1).mean().plot(kind='bar',ax=ax2)
title('South Island average FK costs')
ylabel('$')
```

```
<matplotlib.text.Text at 0x25a83610>
```

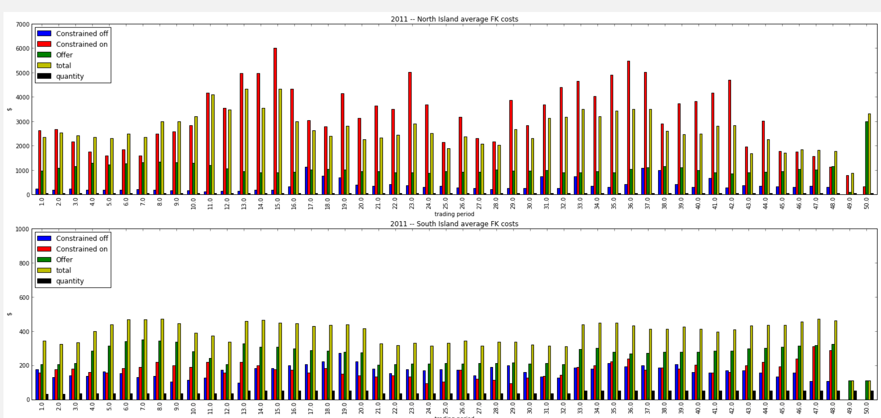


0.5.3 2011

In 2011 NI FK costs were over 5 times those of the SI.

```
fig=plt.figure(17,figsize=[27,12])
ax1=fig.add_subplot(211)
total_ni[total_ni.index.map(lambda x: x[0].year==2011)].groupby(level=1).mean().plot(
    kind='bar',ax=ax1)
title('2011 -- North Island average FK costs')
ylabel('$')
ylim([0,7000])
ax2=fig.add_subplot(212)
total_si[total_si.index.map(lambda x: x[0].year==2011)].groupby(level=1).mean().plot(
    kind='bar',ax=ax2)
title('2011 -- South Island average FK costs')
ylabel('$')
ylim([0,1000])
```

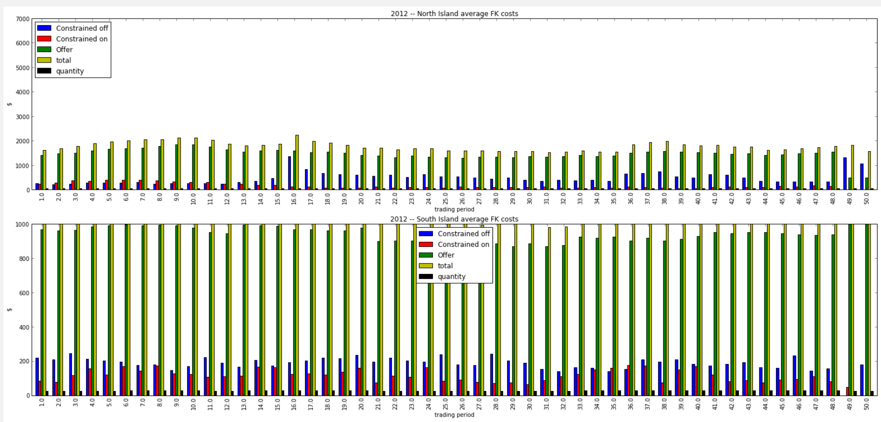
```
(0, 1000)
```



0.5.4 2012

```
fig=plt.figure(18,figsize=[27,12])
ax1=fig.add_subplot(211)
total_ni[total_ni.index.map(lambda x: x[0].year==2012)].groupby(level=1).mean().plot(
    kind='bar',ax=ax1)
title('2012 -- North Island average FK costs')
ylabel('$')
ylim([0,7000])
ax2=fig.add_subplot(212)
total_si[total_si.index.map(lambda x: x[0].year==2012)].groupby(level=1).mean().plot(
    kind='bar',ax=ax2)
title('2012 -- South Island average FK costs')
ylabel('$')
ylim([0,1000])
```

```
(0, 1000)
```



This illustrates the dramatic improvement in FK costs between 2011 and 2012.

There appears to be a bias in the constrained on payments for the NI during the early hours of the morning.
Does this indicate a dc bias in the NI forecast?, i.e., if the forecast is low in the early hours of the morning, then perhaps more FK is needed due to less dispatched generation than what was required?

0.6 NI FK market by Trader

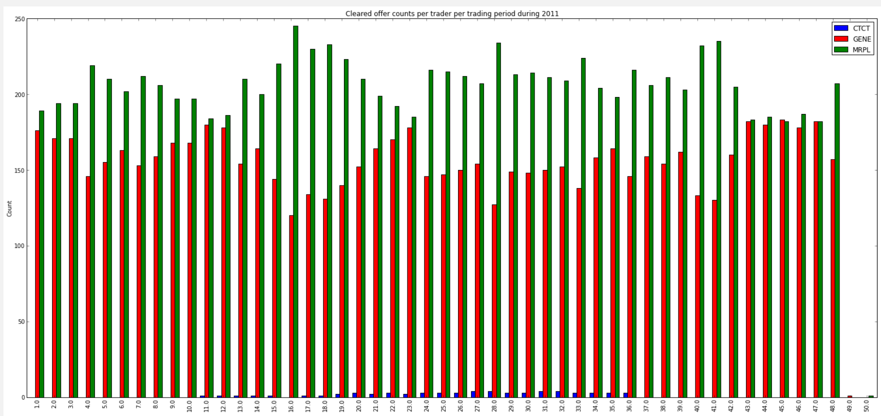
0.6.1 Offer counts per TP

```
year=2011

fig=plt.figure(19,figsize=[27,12])
ax1=fig.add_subplot(111)

nicounts = DataFrame({'GENE':total_ni[total_ni.index.map(lambda x: x[0].year==year)][
    total_ni['Trader']=='GENE'].groupby(level=1).count().Offer, \
```

```
                    ’MRPL’:total_ni[total_ni.index.map(lambda x: x[0].year==year)][
                        total_ni[’Trader’]==’MRPL’].groupby(level=1).count().Offer, \
                    ’CTCT’:total_ni[total_ni.index.map(lambda x: x[0].year==year)][
                        total_ni[’Trader’]==’CTCT’].groupby(level=1).count().Offer}).
                        fillna(0)

nicounts.plot(kind=’bar’,ax=ax1)
title(’Cleared offer counts per trader per trading period during ’ + str(year))
ylabel(’Count’)
```

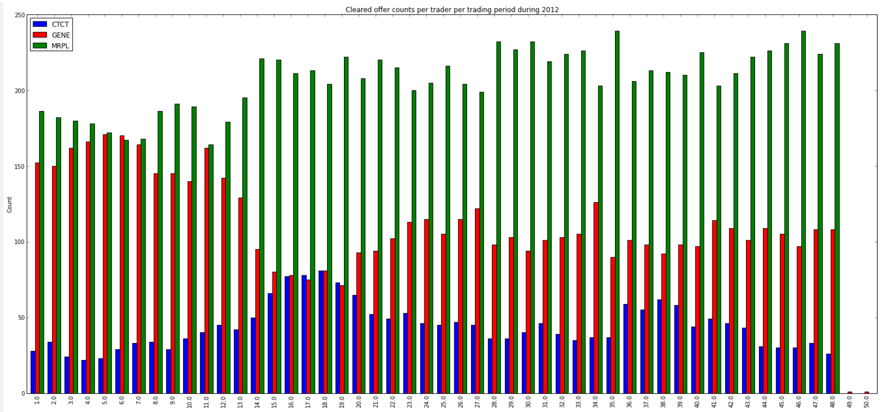
```
/usr/local/lib/python2.7/dist-packages/pandas-0.11.0.dev_14a04dd-py2.7-linux-x86_64.egg/pandas/core/fram
  "DataFrame index.", UserWarning)
```

```
<matplotlib.text.Text at 0x25f35390>
```




```
year=2012

fig=plt.figure(20,figsize=[27,12])
ax1=fig.add_subplot(111)

nicounts = DataFrame({’GENE’:total_ni[total_ni.index.map(lambda x: x[0].year==year)][
    total_ni[’Trader’]==’GENE’].groupby(level=1).count().Offer, \
                    ’MRPL’:total_ni[total_ni.index.map(lambda x: x[0].year==year)][
                        total_ni[’Trader’]==’MRPL’].groupby(level=1).count().Offer, \
                    ’CTCT’:total_ni[total_ni.index.map(lambda x: x[0].year==year)][
                        total_ni[’Trader’]==’CTCT’].groupby(level=1).count().Offer}).
                        fillna(0)

nicounts.plot(kind=’bar’,ax=ax1)
title(’Cleared offer counts per trader per trading period during ’ + str(year))
ylabel(’Count’)
```

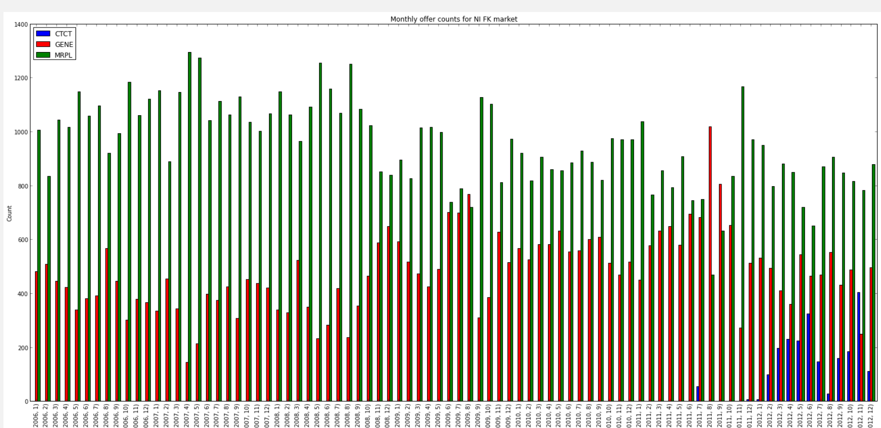
```
<matplotlib.text.Text at 0x22f7c650>
```



So MRPL and CTCT have been cleared more often than GENE in 2012. Lets do counts on a monthly basis to see any trends...

```
years=[2006,2007,2008,2009,2010,2011,2012]
GENE=total_ni[total_ni.index.map(lambda x: (x[0].year in years))][total_ni['Trader']=='
    GENE'].groupby(total_ni[total_ni.index.map(lambda x: (x[0].year in years))][total_ni[
    'Trader']=='GENE'].index.map(lambda x: (x[0].year,x[0].month))).count().Offer
MRPL=total_ni[total_ni.index.map(lambda x: (x[0].year in years))][total_ni['Trader']=='
    MRPL'].groupby(total_ni[total_ni.index.map(lambda x: (x[0].year in years))][total_ni[
    'Trader']=='MRPL'].index.map(lambda x: (x[0].year,x[0].month))).count().Offer
CTCT=total_ni[total_ni.index.map(lambda x: (x[0].year in years))][total_ni['Trader']=='
    CTCT'].groupby(total_ni[total_ni.index.map(lambda x: (x[0].year in years))][total_ni[
    'Trader']=='CTCT'].index.map(lambda x: (x[0].year,x[0].month))).count().Offer
fig=plt.figure(21,figsize=[27,12])
ax1=fig.add_subplot(111)
DataFrame({'GENE':GENE,'MRPL':MRPL,'CTCT':CTCT}).plot(kind='bar',ax=ax1)
title('Monthly offer counts for NI FK market')
ylabel('Count')
```

```
<matplotlib.text.Text at 0x26580fd0>
```

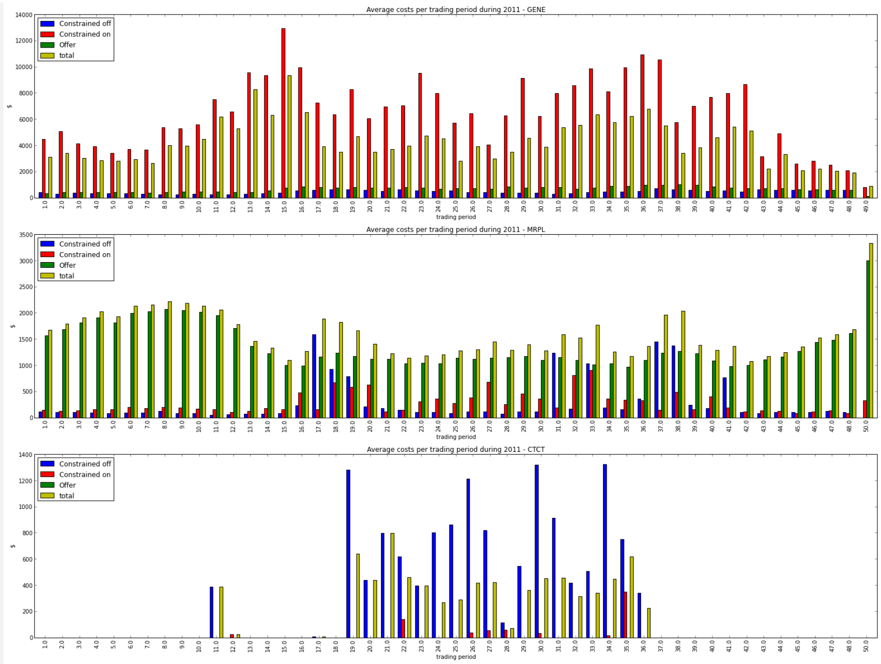


Here we see CTCT entering the NI FK market at the end of 2011. Lets look at costs.

0.6.2 Ave. costs per trading period

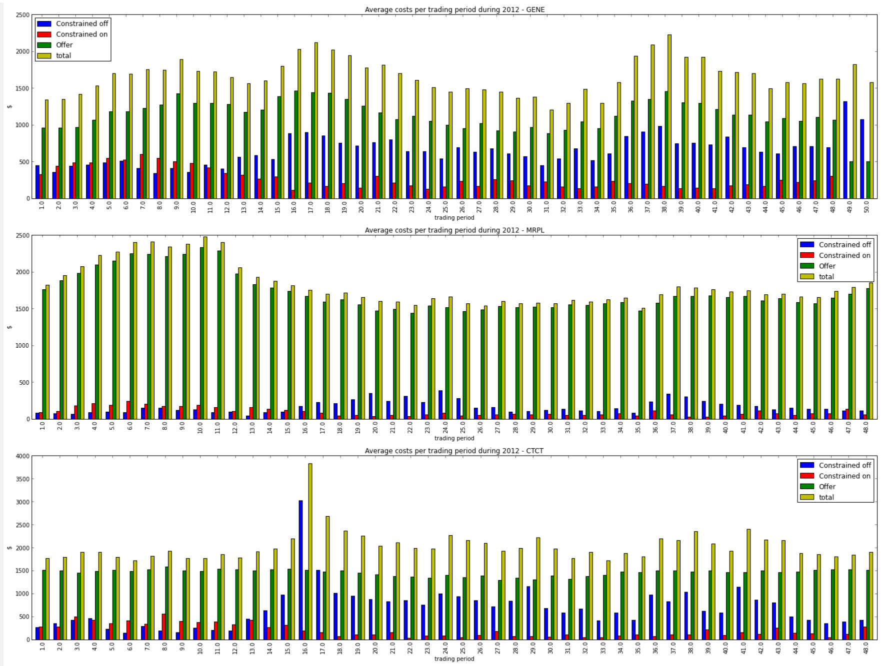
```
year=2011
fig=plt.figure(22,figsize=[27,27])
ax1=fig.add_subplot(411)
gene=total_ni[total_ni.index.map(lambda x: x[0].year==year)][total_ni['Trader']=='GENE'
    ].groupby(level=1).mean()
del gene['quantity']
gene.plot(kind='bar',ax=ax1)
title('Average costs per trading period during ' + str(year) + ' - GENE')
ylabel('$')
ax2=fig.add_subplot(412)
mrpl=total_ni[total_ni.index.map(lambda x: x[0].year==year)][total_ni['Trader']=='MRPL'
    ].groupby(level=1).mean()
del mrpl['quantity']
mrpl.plot(kind='bar',ax=ax2)
title('Average costs per trading period during ' + str(year) + ' - MRPL')
ylabel('$')
ax3=fig.add_subplot(413)
ctct=total_ni[total_ni.index.map(lambda x: x[0].year==year)][total_ni['Trader']=='CTCT'
    ].groupby(level=1).mean().reindex_like(mrpl)
ctct.plot(kind='bar',ax=ax3)
title('Average costs per trading period during ' + str(year) + ' - CTCT')
ylabel('$')
total_ni.reindex_like
```

```
<bound method DataFrame.reindex_like of <class 'pandas.core.frame.DataFrame'>
MultiIndex: 130078 entries, (2005-09-01 00:00:00, 1.0) to (2013-01-31 00:00:00, 48.0)
Data columns:
Constrained off    47369  non-null values
Constrained on     69725  non-null values
Offer              130065  non-null values
Trader             130065  non-null values
total              130066  non-null values
quantity           130065  non-null values
dtypes: float64(5), object(1)>
```

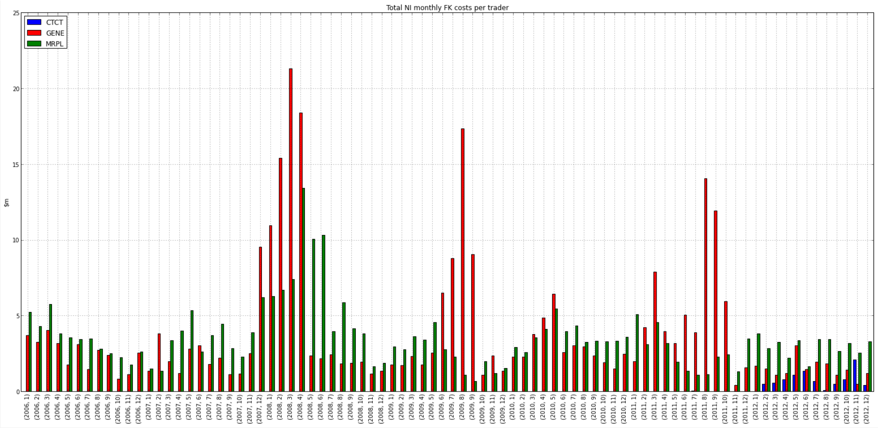



```
year=2012
fig=plt.figure(23,figsize=[27,27])
ax1=fig.add_subplot(411)
gene=total_ni[total_ni.index.map(lambda x: x[0].year==year)][total_ni['Trader']=='GENE'
    ].groupby(level=1).mean()
del gene['quantity']
gene.plot(kind='bar',ax=ax1)
title('Average costs per trading period during ' + str(year) + ' - GENE')
ylabel('$')
ax2=fig.add_subplot(412)
mrpl=total_ni[total_ni.index.map(lambda x: x[0].year==year)][total_ni['Trader']=='MRPL'
    ].groupby(level=1).mean()
del mrpl['quantity']
mrpl.plot(kind='bar',ax=ax2)
title('Average costs per trading period during ' + str(year) + ' - MRPL')
ylabel('$')
ax3=fig.add_subplot(413)
ctct=total_ni[total_ni.index.map(lambda x: x[0].year==year)][total_ni['Trader']=='CTCT'
    ].groupby(level=1).mean()
del ctct['quantity']
ctct.plot(kind='bar',ax=ax3)
title('Average costs per trading period during ' + str(year) + ' - CTCT')
ylabel('$')
```

```
<matplotlib.text.Text at 0x25007dd0>
```




```
years=[2006,2007,2008,2009,2010,2011,2012]
GENE=total_ni[total_ni.index.map(lambda x: (x[0].year in years))][total_ni['Trader']=='
    GENE'].groupby(total_ni[total_ni.index.map(lambda x: (x[0].year in years))][total_ni[
    'Trader']=='GENE'].index.map(lambda x: (x[0].year,x[0].month))).sum()
MRPL=total_ni[total_ni.index.map(lambda x: (x[0].year in years))][total_ni['Trader']=='
    MRPL'].groupby(total_ni[total_ni.index.map(lambda x: (x[0].year in years))][total_ni[
    'Trader']=='MRPL'].index.map(lambda x: (x[0].year,x[0].month))).sum()
CTCT=total_ni[total_ni.index.map(lambda x: (x[0].year in years))][total_ni['Trader']=='
    CTCT'].groupby(total_ni[total_ni.index.map(lambda x: (x[0].year in years))][total_ni[
    'Trader']=='CTCT'].index.map(lambda x: (x[0].year,x[0].month))).sum()
del GENE['quantity']
del MRPL['quantity']
del CTCT['quantity']
GENE=GENE.sum(axis=1)
MRPL=MRPL.sum(axis=1)
CTCT=CTCT.sum(axis=1)


fig=plt.figure(24,figsize=[27,12])
ax1=fig.add_subplot(111)
(DataFrame({'GENE':GENE,'MRPL':MRPL,'CTCT':CTCT})/1000000.0).plot(kind='bar',ax=ax1)
title('Total NI monthly FK costs per trader')
ylabel('$m')
grid('on')
```

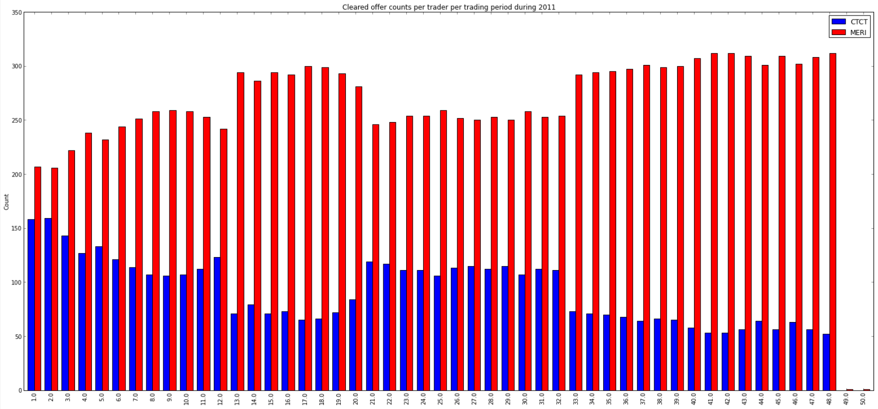


0.7 SI FK market by Trader

0.7.1 Offer counts per trading period

```
year=2011

fig=plt.figure(25,figsize=[27,12])
ax1=fig.add_subplot(111)

sicounts = DataFrame({'MERI':total_si[total_si.index.map(lambda x: x[0].year==year)][
    total_si['Trader']=='MERI'].groupby(level=1).count().Offer, \
                    'CTCT':total_si[total_si.index.map(lambda x: x[0].year==year)][
                        total_si['Trader']=='CTCT'].groupby(level=1).count().Offer}).
                        fillna(0)

sicounts.plot(kind='bar',ax=ax1)
title('Cleared offer counts per trader per trading period during ' + str(year))
ylabel('Count')
```

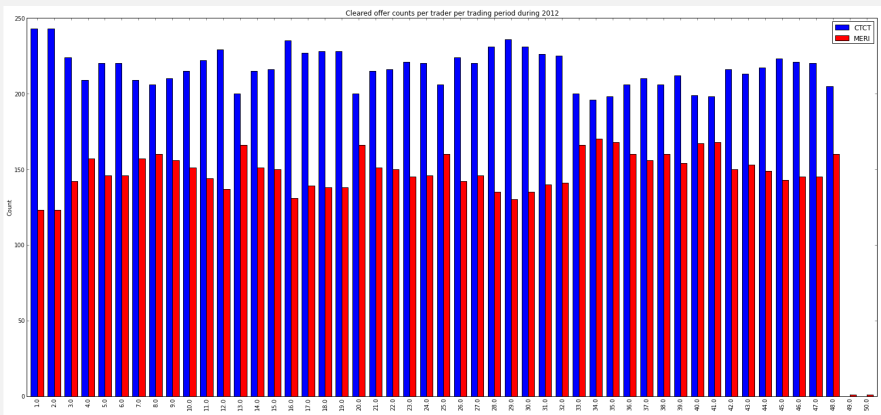
```
<matplotlib.text.Text at 0x25536a90>
```




```
year=2012
```

```
fig=plt.figure(26,figsize=[27,12])
ax1=fig.add_subplot(111)

sicounts = DataFrame({'MERI':total_si[total_si.index.map(lambda x: x[0].year==year)][
    total_si['Trader']=='MERI'].groupby(level=1).count().Offer, \
                    'CTCT':total_si[total_si.index.map(lambda x: x[0].year==year)][
                        total_si['Trader']=='CTCT'].groupby(level=1).count().Offer}).
                        fillna(0)

sicounts.plot(kind='bar',ax=ax1)
title('Cleared offer counts per trader per trading period during ' + str(year))
ylabel('Count')
```

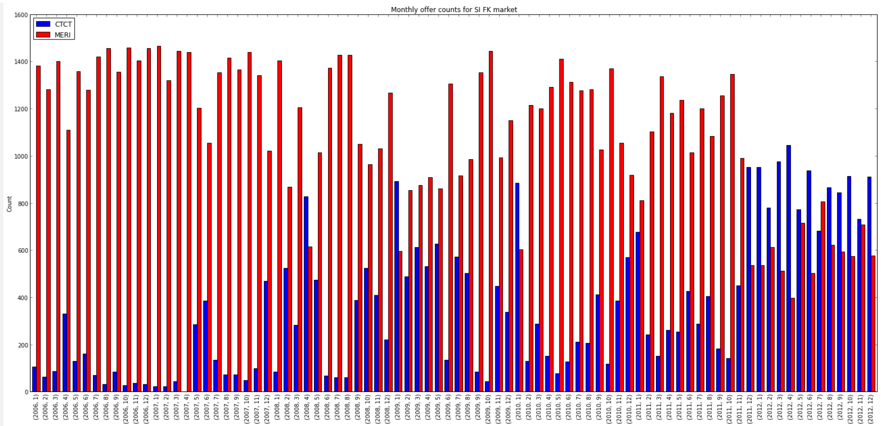
```
<matplotlib.text.Text at 0x2c318d90>
```



This is interesting as we see a market dominated by meridian in 2011 swaped to be dominated by Contact in 2012. Lets group by month and get a time series to see how the counts have changed.

```
years=[2006,2007,2008,2009,2010,2011,2012]
MERI=total_si[total_si.index.map(lambda x: (x[0].year in years))][total_si['Trader']=='
    MERI'].groupby(total_si[total_si.index.map(lambda x: (x[0].year in years))][total_si[
    'Trader']=='MERI'].index.map(lambda x: (x[0].year,x[0].month))).count().Offer
CTCT=total_si[total_si.index.map(lambda x: (x[0].year in years))][total_si['Trader']=='
    CTCT'].groupby(total_si[total_si.index.map(lambda x: (x[0].year in years))][total_si[
    'Trader']=='CTCT'].index.map(lambda x: (x[0].year,x[0].month))).count().Offer
fig=plt.figure(27,figsize=[27,12])
ax1=fig.add_subplot(111)
DataFrame({'MERI':MERI,'CTCT':CTCT}).plot(kind='bar',ax=ax1)
title('Monthly offer counts for SI FK market')
ylabel('Count')
```

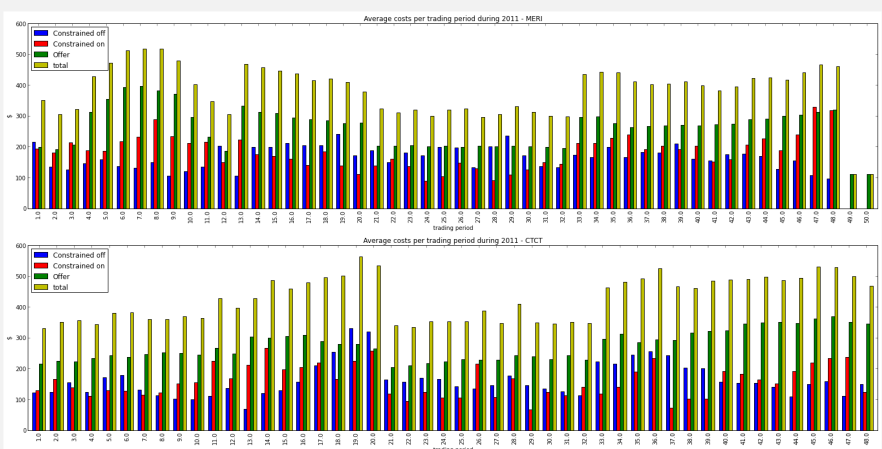
```
<matplotlib.text.Text at 0x2cffddd0>
```



0.7.2 Ave. costs per trading period

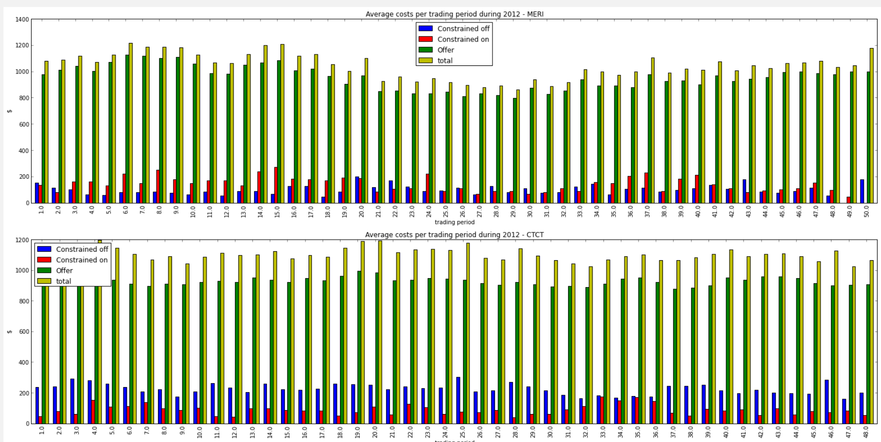
```python
year=2011
fig=plt.figure(28,figsize=[27,27])
ax1=fig.add_subplot(411)
meri=total_si[total_si.index.map(lambda x: x[0].year==year)][total_si['Trader']=='MERI'
    ].groupby(level=1).mean()
del meri['quantity']
meri.plot(kind='bar',ax=ax1)
title('Average costs per trading period during ' + str(year) + ' - MERI')
ylabel('$')
ax2=fig.add_subplot(412)
ctct=total_si[total_si.index.map(lambda x: x[0].year==year)][total_si['Trader']=='CTCT'
    ].groupby(level=1).mean()
del ctct['quantity']
ctct.plot(kind='bar',ax=ax2)
title('Average costs per trading period during ' + str(year) + ' - CTCT')
ylabel('$')
```

```
<matplotlib.text.Text at 0x25424590>
```



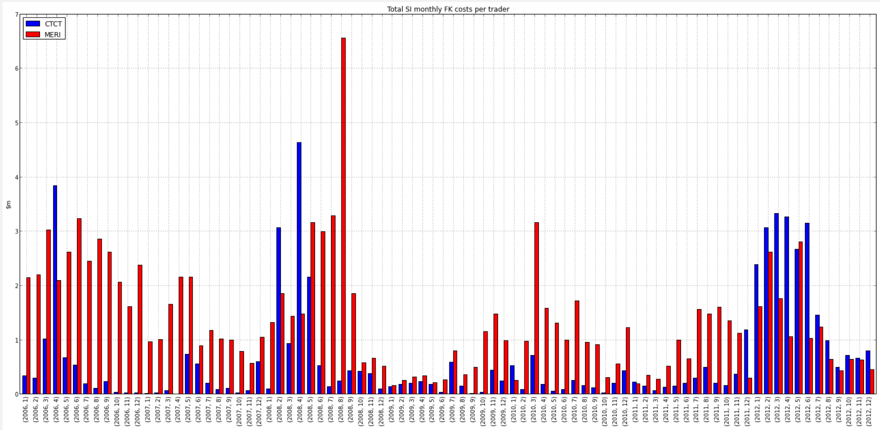
```
year=2012
fig=plt.figure(29,figsize=[27,27])
ax1=fig.add_subplot(411)
meri=total_si[total_si.index.map(lambda x: x[0].year==year)][total_si['Trader']=='MERI'
    ].groupby(level=1).mean()
del meri['quantity']
meri.plot(kind='bar',ax=ax1)
title('Average costs per trading period during ' + str(year) + ' - MERI')
ylabel('$')
ax2=fig.add_subplot(412)
ctct=total_si[total_si.index.map(lambda x: x[0].year==year)][total_si['Trader']=='CTCT'
    ].groupby(level=1).mean()
del ctct['quantity']
ctct.plot(kind='bar',ax=ax2)
title('Average costs per trading period during ' + str(year) + ' - CTCT')
ylabel('$')
```

```
<matplotlib.text.Text at 0x23e64a90>
```




```
years=[2006,2007,2008,2009,2010,2011,2012]
MERI=total_si[total_si.index.map(lambda x: (x[0].year in years))][total_si['Trader']=='
    MERI'].groupby(total_si[total_si.index.map(lambda x: (x[0].year in years))][total_si[
    'Trader']=='MERI'].index.map(lambda x: (x[0].year,x[0].month))).sum()
CTCT=total_si[total_si.index.map(lambda x: (x[0].year in years))][total_si['Trader']=='
    CTCT'].groupby(total_si[total_si.index.map(lambda x: (x[0].year in years))][total_si[
    'Trader']=='CTCT'].index.map(lambda x: (x[0].year,x[0].month))).sum()
del MERI['quantity']
del CTCT['quantity']
MERI=MERI.sum(axis=1)
CTCT=CTCT.sum(axis=1)

fig=plt.figure(30,figsize=[27,12])
ax1=fig.add_subplot(111)
(DataFrame({'MERI':MERI,'CTCT':CTCT})/1000000.0).plot(kind='bar',ax=ax1)
title('Total SI monthly FK costs per trader')
ylabel('$m')
```

```
grid('on')
```



0.8 FK costs output to csv

```
total_ni.to_csv('fk_costs_NI.csv')
total_si.to_csv('fk_costs_SI.csv')
```

```
total_ni
```

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 130078 entries, (2005-09-01 00:00:00, 1.0) to (2013-01-31 00:00:00, 48.0)
Data columns:
Constrained off    47369  non-null values
Constrained on     69725  non-null values
Offer              130065  non-null values
Trader             130065  non-null values
total              130066  non-null values
quantity           130065  non-null values
dtypes: float64(5), object(1)
```

0.9 To do

Need to look at difference in the cleared and metered data for each FK generator

0.10 Appendix

These scripts were used to auto download and process the data from the EA website

File renamer

months='jan':1,'feb':2,'mar':3,'apr':4,'may':5,'jun':6,'jul':7,'aug':8,'sep':9,'oct':10,'nov':11,'dec':12 year='2005' os.chdir(path+year) for f in os.listdir(path + year): if f.split(' $')[1][:2].lower()=='cl':clearedos.rename(f,year+'_'+str(months[f.split('$ $')[3].split('.')[0][:3].lower()]).zfill(2)+'_clearedoffers.xls')iff.split('$ $')[1][:2].$

formatter = logging.Formatter('—consoleLogger = logging.StreamHandler() consoleLogger.setLevel(logging.INFO) consoleLogger.setFormatter(formatter) logging.getLogger(").addHandler(consoleLogger)

fileLogger = logging.handlers.RotatingFileHandler(filename=path+ 'fkdata.log',maxBytes = 1024*1024, backupCount = 9) fileLogger.setLevel(logging.ERROR) fileLogger.setFormatter(formatter) logging.getLogger(").addHandler(fileLogger)

logger = logging.getLogger('FK DATA SCRAPER') logger.setLevel(logging.INFO) logger.info('Automated Frequency Keeping data scraper'.center(126))

fk$_{d}ict =$

def fk$_{n}ame_{v}alue(r) : Getfkfilesiteidname = r.get_{d}ata().split('class = "documentxlsx"')[1].split('")[1].replace('','').replace('< td >','').replace('< spanclass = "grey" >','')[: -12]retur$

url='https://www.ea.govt.nz/industry/pso-cq/system-operations/fk-data/' br = mechanize.Browser() Browser br.set$_{h}andle_{r}efresh(mechanize._{h}ttp.HTTPRefreshProcessor(), max_{t}ime = 1)Followsrefresh0butnothangsonrefresh > 0br.addheaders = [('User - agent','Mozilla/5.0(X11; U; Linuxi686; en - US; rv : 1.9.0.1)Gecko/2008071615Fedora/3.0.1 - 1.fc9Firefox/3.0.1')]User - Agent(thisischeating, ok?)r = br.open(url)$

for f in arange(1,len(fkscrape)): path = fkscrape[f].split('class="full-link" href=')[1].split(' title')[0].replace('"','') file$_{n}ame = fkscrape[f].split('class = "full - link"href =')[1].split('title = "')[1].split('"')[0].replace('Download','')fk_{d}ict[file_{n}ame] = path$

url$_{h}ead =' https : //www.ea.govt.nz'$

for file$_{n}ame, pathinfk_{d}ict.iteritems() :$

print 'Downloading: ' + file$_{n}amewf = open(file_{n}ame, mode = "wb")try : wf_{i}mage = br.open(url_{h}ead + path).read()wf.write(wf_{i}mage)wf.close()except : continue$

month$_{m}ap = 'jan' : 1,'feb' : 2,'mar' : 3,'apr' : 4,'may' : 5,'jun' : 6,'jul' : 7,'aug' : 8,'sep' : 9,'oct' : 10,'nov' :$

for f in os.listdir(path): f2 = f.replace('-','').lower() if f2[0:2]=='fk': if 'clearedoffers' in f2: print f2.split('.')[0][-5:-2] rename files top sensible names! newname = '20' + f2.split('.')[0][-2:] + '$'+str(month_{m}ap[f2.split('.')[0][-5:-2]]).zfill(2)+'_{c}learedoffers.'+f2.split('.')[1]os.rename(f,newname)$

month$_{m}ap = 'jan' : 1,'feb' : 2,'mar' : 3,'apr' : 4,'may' : 5,'jun' : 6,'jul' : 7,'aug' : 8,'sep' : 9,'oct' : 10,'nov' :$

for f in os.listdir(path): f2 = f.replace('-','').lower() if f2[0:2]=='fk': if 'fkconstrainedcosts' in f2: print f2.split('.')[0][-5:-2] rename files top sensible names! newname = '20' + f2.split('.')[0][-2:] + '$'+str(month_{m}ap[f2.split('.')[0][-5:-2]]).zfill(2)+'_{c}onstrainedcosts.'+f2.split('.')[1]print'Renaming'+f+'--to-->'+newnameos.rename(f,$