A Test of Replacement Level

I was browsing twitter and I saw a series of tweets made by Tom Tango and Mitchel Lichtman about realistic manifestations of replacement level. In one of the tweets, Tango linked to an ESPN page and stated, "Here are the players who signed a minor league free agent contract last season. I'd love for someone to total up their stats, including WAR, PA, and IP, to see how well rWAR and fWAR held up." I decided to test this.

I went to ESPN's record of free agent signings, which goes back to the 2006 offseason, and compiled all the minor league free agent signings from the website. I then collected from FanGraphs and Baseball-Reference, respectively, the pitcher and hitter data from the 2007-2017 seasons that contained the relevant data, namely WAR, IP, and PA. For all the minor league free agents that played in MLB the following season, I recorded for that season their fWAR and bWAR (pitchers' offense included) as well as the IP for pitchers and PA for both pitchers (if applicable) and hitters.

Now for the results: I calculated the sum of PA, IP and fWAR and bWAR for each player in each of those seasons. There were 328 players in the sample that compiled 25,096 PA and 7541 IP. The sums of fWAR and bWAR were around 51 and 55 WAR, respectively. Taking the fWAR and bWAR per player, the mean is about .16 WAR per player, with closer to .11 for position players and .2 for pitchers. About 13-14% of players compiled 1 WAR or greater.

What I expected was that the sums of each WAR construction would total closer to 0 WAR. Considering that the mean WAR per player is close to replacement level, the result is not wholly dissatisfying, but overall still may leave an explanation to be desired. One explanation is that there remains the possibility of some selection bias in the data. Obviously, the minor league free agents that were picked to play meaningful MLB innings were ones deemed by MLB teams to be the more promising of the crop, and any players who did not perform to teams' expected baseline were removed quickly and were never given an extended chance (or any chance at all). Furthermore, some of them may be players who were originally undervalued by the industry and forced to sign minor league deals, possibly because of poor performance the previous year or two due to random variation or injuries. A comparison of these players with their preseason projections would therefore be in order for clarification. Additionally, using two years of data instead of just the year afterward or a larger pool of players should also lead further away from potential bias. Another more unlikely explanation is that there is a flaw with the established replacement level.

Further research – Lichtman proposed comparing Steamer or ZIPS projections of these players to their actual performance. For the reasons that I alluded to earlier, since some players' projections

were well above replacement level, and they performed as such, teams may have undervalued them, and if some players' projections were well below replacement level, then teams may have overvalued them.

The method: I downloaded csv files from FG and BB-Ref for the seasonal data, and I copied and pasted the tables from the ESPN website (tedious but faster than figuring out how to scrape the data by my estimation). I used python pandas with jupyter notebook. Here are some sample screenshots. The github repo is located at the following URL: https://github.com/rontsung/replevel-test

In [1]:
```
# dependencies

import pandas as pd
import json
import datetime
```

In [164]:
```
data = {}
for i in range(2006,2017,1):
    # filter out the FA that signed minor league deals
    fa = pd.read_csv(str(i)+"FA.csv")
    fa = fa[fa["PLAYER"]!="PLAYER"]
    fa = fa[fa["DOLLARS"]=="Minor Lg"]
    # compile all csv files into one dict - FA, FG and bbref batters and pitchers
    fbat = pd.read_csv(str(i+1)+" bleaders.csv")
    fpitch = pd.read_csv(str(i+1)+" pleaders.csv")
    bbat = pd.read_csv(str(i+1)+" bWAR batters.csv")
    bpitch = pd.read_csv(str(i+1)+" bWAR pitchers.csv")
    data[i] = [fa,{"FG":[fbat,fpitch], "BBREF":[bbat, bpitch]}]
data
```

Out[164]:
```
{2006: [              PLAYER POS AGE  STATUS    2015 TEAM    NEW TEAM  YRS  RK    DOLLARS
  0   Sandy Alomar Jr.   C  51  Signed    White Sox        Mets  NaN  NR  Minor Lg
  1        Rick Ankiel  CF  38  Signed    Cardinals  Cardinals  NaN  NR  Minor Lg
  4    Ronnie Belliard  2B  42  Signed    Cardinals  Nationals  NaN  NR  Minor Lg
  8         Bruce Chen  SP  40  Signed      Orioles    Rangers  NaN  NR  Minor Lg
  9        Mike DeJean  RP  47  Signed      Rockies    Rockies  NaN  NR  Minor Lg
  10        Einar Diaz   C  45  Signed      Dodgers    Pirates  NaN  NR  Minor Lg
  11     Mike DiFelice   C  48  Signed         Mets       Mets  NaN  NR  Minor Lg
  12       Shawn Estes  SP  45  Signed       Padres     Padres  NaN  NR  Minor Lg
```

In [135]:
```
# create dict with all players for each year - both their bWAR and fWAR as well as their PA and IP if applicable

a = datetime.datetime.now()
players = {}
for e in data:
    cur = e+1
    players[cur] = [{}]
    fadata = data[e][0]
    seadata = data[e][1]
    # run through each free agent
    for index1, row1 in fadata.iterrows():
        fa = row1["PLAYER"]
        for site in seadata:
            k = 0
            # run through the seasonal data
            for f in seadata[site]:
                for index2, row2 in f.iterrows():
                    # identify the free agents that played in MLB that year
                    if fa[:fa.find(" ")] in row2["Name"] and fa[fa.find(" ")+1:] in row2["Name"] or fa == row2["Name"]:
                        if fa not in players[cur][0]:
                            players[cur][0][fa] = {"fWAR":0.0, "bWAR":0.0}
                        war = row2["WAR"]
                        # store the data in the dict
                        if site == "FG":
                            players[cur][0][fa]["fWAR"] += war
                            if k == 1:
                                players[cur][0][fa]["IP"] = row2["IP"]
                            else:
                                players[cur][0][fa]["PA"] = row2["PA"]
                        else:
                            if row2["WAR"] == row2["WAR"]:
                                players[cur][0][fa]["bWAR"] += war
                k+=1
b = datetime.datetime.now()
print(b-a)
```

```
0:05:32.115728
```

In [137]:
```
players
```

Out[137]:
```
{2007: [{'Aaron Sele': {'IP': 53.2, 'PA': 6, 'bWAR': -0.3, 'fWAR': 0.0},
   'Brian Moehler': {'IP': 59.2, 'PA': 5, 'bWAR': 0.6, 'fWAR': 0.0},
   'Bruce Chen': {'IP': 10.0, 'PA': 0, 'bWAR': -0.3, 'fWAR': -0.2},
   'Dan Kolb': {'IP': 3.0, 'PA': 0, 'bWAR': -0.1, 'fWAR': -0.1},
   'David Newhan': {'PA': 83, 'bWAR': -0.4, 'fWAR': 0.0},
   'Eddie Guardado': {'IP': 13.2, 'PA': 0, 'bWAR': -0.3, 'fWAR': 0.1},
   'Jamey Wright': {'IP': 77.0, 'PA': 2, 'bWAR': 1.4, 'fWAR': 0.4},
   'John Mabry': {'PA': 39, 'bWAR': -0.3, 'fWAR': -0.3},
   'Jon Knott': {'PA': 19, 'bWAR': 0.0, 'fWAR': 0.0},
   'Mark Redman': {'IP': 41.1,
    'PA': 14,
    'bWAR': -0.8999999999999999,
    'fWAR': 0.19999999999999998},
   'Matt Herges': {'IP': 48.2,
    'PA': 6,
    'bWAR': 1.0999999999999999,
    'fWAR': 0.30000000000000004},
   'Matt Stairs': {'PA': 405, 'bWAR': 2.4, 'fWAR': 1.4},
   'Mike DiFelice': {'PA': 47, 'bWAR': 0.0, 'fWAR': 0.1},
```

In [153]:
```
# for each year, add all the IP, PA, bWAR, and fWAR and store in dict

totals = {}
for e in players:
    totals[e] = {"Total IP": 0, "Total PA": 0, "Total fWAR": 0, "Total bWAR": 0}
    for p in players[e][0]:
        totals[e]["Total fWAR"] += players[e][0][p]["fWAR"]
        totals[e]["Total bWAR"] += players[e][0][p]["bWAR"]
        if "PA" in players[e][0][p]:
            totals[e]["Total PA"] += players[e][0][p]["PA"]
        if "IP" in players[e][0][p]:
            IP = players[e][0][p]["IP"]
            if ".1" in str(IP):
                IP += .23
            elif ".2" in str(IP):
                IP += .47
            totals[e]["Total IP"] += IP
totals
```

In [154]:
```
# for all the years, add the fWAR, bWAR, PA, and IP and store in dict

totalfwar = 0
totalbwar = 0
totalpa = 0
totalip = 0
for e in totals:
    totalfwar += totals[e]["Total fWAR"]
    totalbwar += totals[e]["Total bWAR"]
    totalpa += totals[e]["Total PA"]
    totalip += totals[e]["Total IP"]
fulltotal = {"Total fWAR": totalfwar, "Total bWAR": totalbwar, "Total PA": totalpa, "Total IP": totalip}
print(fulltotal)
```

```
{'Total fWAR': 50.800000000000004, 'Total bWAR': 55.00000000000001, 'Total PA': 25096, 'Total IP': 7541.03}
```

In [177]:
```
# calculate the WAR/player in several varieties, e.g. fWAR per player

fWARbatters = 0
fWARpitchers = 0
bWARpitchers = 0
bWARpitchers = 0
batters = 0
pitchers = 0
totalplayers = 0
for e in players:
    for f in players[e]:
        totalplayers += len(f)
for e in players:
    for p in players[e][0]:
        if "IP" in players[e][0][p]:
            pitchers += 1
            try:
                fWARpitchers += players[e][0][p]["fWAR"]
                bWARpitchers += players[e][0][p]["bWAR"]
            except:
                print("")
batters = totalplayers - pitchers
fWARbatters = totalfwar - fWARpitchers
bWARbatters = totalbwar - bWARpitchers
print(totalfwar/totalplayers, totalbwar/totalplayers, fWARbatters/batters, bWARbatters/batters, fWARpitchers/pitchers, b
print(fWARbatters*600/totalpa, bWARbatters*600/totalpa)
```

```
0.15487804878048783 0.1676829268292683 0.12606060606060607 0.10424242424242422 0.18404907975460125 0.2319018404907976
2
0.49729040484539366 0.4112209116990755
```