Run Expectancy & wRC+

Isabelle Perez

Import Data & Necessary Packages

```
import pandas as pd
import numpy as np
```

```
# read in data

data = pd.read_csv('../data/synergy/run_expectancy.csv')
hr_data = pd.read_csv('../data/synergy/hr_expectancy.csv')
single_data = pd.read_csv('../data/synergy/1b_expectancy.csv')
double_data = pd.read_csv('../data/synergy/2b_expectancy.csv')
triple_data = pd.read_csv('../data/synergy/3b_expectancy.csv')
hbp_data = pd.read_csv('../data/synergy/hbp_expectancy.csv')
bb_data = pd.read_csv('../data/synergy/bb_expectancy.csv')
fall_stats = pd.read_csv('../data/synergy/Fall Stats(Sheet1).csv')
bigeast_stats = pd.read_csv('../data/synergy/bigeast_stats(Sheet1).csv')
```

Necessary Functions

```
def summarize_bases(row):
    '''returns base number if player is on said base, else '''
    return (f"{'1' if row['on_1b'] else '_'}"
            f"{'2' if row['on_2b'] else '_'}"
            f"{'3' if row['on_3b'] else '_'}")

def get_table(data):
    '''return expectancy table for specific data frame'''
```

```
    exp_table = data.groupby(['outs', 'base_state'])
    exp_table = exp_table['runs'].mean().reset_index()
    exp_table = exp_table.pivot_table(
                    index = 'base_state',
                    columns = 'outs',
                    values = 'runs',
                    aggfunc = 'mean')

    return exp_table.fillna(0)

def start_re(table, data, run_exp):
    '''return starting run expectancy for event'''
    event_re = table.multiply(run_exp).fillna(0)
    event_re = event_re.values.sum() / len(data)

    return event_re.round(3)
```

Run Expectancies

Overall Run Expectancy

```
# summarize data for runners on base
data['base_state'] = data.apply(summarize_bases, axis = 1)

# group by combinations of outs and runners on base
run_expectancy = data.groupby(['outs', 'base_state'])

# return to data frame structure
run_expectancy = run_expectancy['runs_scored'].mean().reset_index()

# pivot such that outs are columns
run_exp = run_expectancy.pivot_table(
            index = 'base_state',
            columns = 'outs',
            values = 'runs_scored',
            aggfunc = 'mean')

# format numbers to three decimal places
pd.options.display.float_format = '{:.3f}'.format
```

```
run_exp
```

outs base_state	0	1	2
123	1.400	1.100	0.533
12_	0.231	0.288	0.431
1_3	0.400	0.783	0.429
1__	0.075	0.129	0.140
_23	0.750	0.893	0.600
_2_	0.149	0.256	0.250
__3	0.400	0.478	0.314
___	0.035	0.036	0.042

Homerun

```
# average run expectancy for homeruns
hr_avg = sum(hr_data['runs']) / len(hr_data)

# get expectancy table
hr_exp_table = get_table(hr_data)

# get starting home run expectancy
hr_start_re = start_re(hr_exp_table, hr_data, run_exp)

# final hr run expectancy
hr_re = hr_avg - hr_start_re

hr_re
```

```
np.float64(1.454090909090909)
```

Single

```
# average run expectancy for homeruns
single_avg = sum(single_data['runs']) / len(single_data)
```

```
# get expectancy table
single_exp_table = get_table(single_data)

# get starting home run expectancy
single_start_re = start_re(single_exp_table, single_data, run_exp)

# final hr run expectancy
single_re = single_avg - single_start_re

single_re
```

```
np.float64(0.2935813953488372)
```

Double

```
# average run expectancy for homeruns
double_avg = sum(double_data['runs']) / len(double_data)

# get expectancy table
double_exp_table = get_table(double_data)

# get starting home run expectancy
double_start_re = start_re(double_exp_table, double_data, run_exp)

# final hr run expectancy
double_re = double_avg - double_start_re

double_re
```

```
np.float64(0.34720338983050847)
```

Triple

```
# average run expectancy for homeruns
triple_avg = sum(triple_data['runs']) / len(triple_data)

# get expectancy table
```

```
triple_exp_table = get_table(triple_data)

# get starting home run expectancy
triple_start_re = start_re(triple_exp_table, triple_data, run_exp)

# final hr run expectancy
triple_re = triple_avg - triple_start_re

triple_re
```

```
np.float64(0.4022857142857143)
```

Hit by pitch

```
# average run expectancy for homeruns
hbp_avg = sum(hbp_data['runs']) / len(hbp_data)

# get expectancy table
hbp_exp_table = get_table(hbp_data)

# get starting home run expectancy
hbp_start_re = start_re(hbp_exp_table, hbp_data, run_exp)

# final hr run expectancy
hbp_re = hbp_avg - hbp_start_re

hbp_re
```

```
np.float64(0.021707865168539325)
```

Walk

```
# average run expectancy for homeruns
bb_avg = sum(bb_data['runs']) / len(bb_data)

# get expectancy table
bb_exp_table = get_table(bb_data)
```

```
# get starting home run expectancy
bb_start_re = start_re(bb_exp_table, bb_data, run_exp)

# final hr run expectancy
bb_re = bb_avg - bb_start_re

bb_re
```

```
np.float64(0.013833333333333333)
```

Calculate WRC+

```
# big east 2024 stats
league_data = pd.DataFrame({'BB': [1037], 'HBP': [334], 'X1B': [1327],
                                    'X2B': [397], 'X3B': [53], 'HR': [253],
                                    'AB': [7270], 'SF': [89]})

def woba(player):
    '''return woba for given player'''
    numerator = ((0.0138 * player['BB']) + (0.0217 * player['HBP'])
                    + (0.2936 * player['X1B']) + (0.3472 * player['X2B'])
                    + (0.4023 * player['X1B']) + (1.4541 * player['HR']))
    denominator = player['AB'] + player['BB'] + player['SF'] + player['HBP']

    woba = numerator / denominator

    return woba

def wraa(player):
    '''return wraa for given player'''
    player_woba = woba(player)
    league_woba = woba(league_data)

    wraa = (player_woba - league_woba) * player['PA']

    return wraa

def wrc(player):
    '''return wrc for given player'''
```

```
    player_woba = woba(player)
    league_woba = woba(league_data)

    wrc = ((player_woba - league_woba) + (1462 / 8836)) * 8836

    return wrc

bigeast_wrc = 0

for index, row in bigeast_stats.iterrows():
    cur_player = bigeast_stats.iloc[index]
    bigeast_wrc += wrc(cur_player)

def wrc_plus(player):
    '''return wrc+ for given player'''
    numerator = ((wraa(player) / player['PA']) + (1462 / 8836)
                 + (1462 / 8836) - (100 * (1462 / 8836)))
    denominator = bigeast_wrc / 8836

    return numerator / denominator
```

Fall 2024 Stats

```
wrc_dict = {}

for index, row in fall_stats.iterrows():
    cur_player = fall_stats.iloc[index]
    wrc_dict[str(cur_player['First'] + ' ' + cur_player['Last'])] = wrc_plus(cur_player)

# convert to data frame and rename columns
wrc_data = pd.DataFrame(wrc_dict).T.reset_index()
wrc_data = wrc_data.rename(columns = {'index': 'name', 0: 'wRC+'})

# sort by best to worst wRC+
wrc_data.sort_values(by = 'wRC+', ascending = False, inplace = True)

wrc_data
```

	name	wRC+
2	Tyler Minick	-1.359
1	Cayden Suchy	-1.362
5	Matt Garbowski	-1.363
0	Aidan Dougherty	-1.363
7	Beau Root	-1.365
8	Grant MacArthur	-1.367
4	Caleb Shpur	-1.367
12	Maddix Dalena	-1.368
10	Mike Oates	-1.368
11	Drew Kron	-1.369
3	Sam Biller	-1.369
14	Gabriel Tirado	-1.369
6	Bryan Padilla	-1.370
16	Connor Lane	-1.371
9	Jack LaRose	-1.372
13	Rob Rispoli	-1.373
17	Anthony Belisario	-1.373
15	Carter Groen	-1.373
18	Ryan Daniels	-1.378

Spring 2024 Stats

```
first = ['Bryan', 'Caleb', 'Drew', 'Maddix', 'Tyler', 'Matt', 'Ryan']
last = ['Padilla', 'Shpur', 'Kron', 'Dalena', 'Minick', 'Garbowski', 'Daniels']
bb = [28, 23, 3, 27, 4, 24, 10]
hbp = [3, 11, 1, 8, 4, 2, 1]
ab = [195, 194, 40, 288, 139, 154, 57]
x1b = [33, 34, 5, 21, 20, 33, 6]
x2b = [14, 14, 2, 12, 7, 6, 2]
x3b = [1, 0, 0, 0, 1, 0, 0]
hr = [6, 3, 0, 13, 9, 4, 3]
sf = [2, 3, 0, 1, 1, 1, 0]
pa = [232, 233, 45, 225, 149, 185, 71]

spring_stats = pd.DataFrame({'First': first, 'Last': last, 'BB': bb,
                             'HBP': hbp, 'AB': ab, 'X1B': x1b, 'X2B': x2b,
                             'X3B': x3b, 'HR': hr, 'SF': sf, 'PA': pa})
```

```
wrc_dict_2 = {}

for index, row in spring_stats.iterrows():
    cur_player = spring_stats.iloc[index]
    wrc_dict_2[str(cur_player['First'] + ' ' + cur_player['Last'])] = wrc_plus(cur_player)

# convert to data frame and rename columns
wrc_data_2 = pd.DataFrame(wrc_dict_2).T.reset_index()
wrc_data_2 = wrc_data_2.rename(columns = {'index': 'name', 0: 'wRC+'})

# sort by best to worst wRC+
wrc_data_2.sort_values(by = 'wRC+', ascending = False, inplace = True)

wrc_data_2
```

	name	wRC+
4	Tyler Minick	-1.366
5	Matt Garbowski	-1.368
0	Bryan Padilla	-1.369
1	Caleb Shpur	-1.370
6	Ryan Daniels	-1.371
3	Maddix Dalena	-1.373
2	Drew Kron	-1.375