Clinical Psychology application

May 2, 2021

1 UK Clinical Psychology application. How to increase your chances?

In this project, we are going to explore the Clinical Psychology training application statistics. The Doctorate in Clinical Psychology is a three-year programme of academic and clinical training offered by different universities across the UK. The programme aims to train practitioners and covers a wide geographical area.

Each applicant can apply to up to 4 NHS universities for funded courses. Self-funded courses are also available but no limit is established.

The Clinical Psychology training is a highly competitive programme, and this analysis aims to identify strategies for applicants to enhance their chances of being accepted by one of the universities.

The plan for this project is to deploy an interactive dashboard to check universities individually with Plotly and Dash on Heroku. Furthermore, an article will be published in Medium showing the main findings.

All the data were obtained on the clearing house website: - https://www.leeds.ac.uk/chpccp/numbers.html

[1]:
```
import pandas as pd
import plotly.offline as pyo
import plotly.graph_objs as go
%matplotlib inline

import plotly.io as pio
pio.renderers.default = "notebook+pdf"

# pyo.init_notebook_mode(connected=True) # To visualise Plotly on Jupyter␣
↪Notebook
```

[2]:
```
# Dataframe containing overall results for funded courses

overall = [["2005", 2125, 7961, 588, "28%"],
           ["2006", 2442, 9152, 554, "23%"],
           ["2007", 2346, 8973, 582, "25%"],
           ["2008", 2323, 8566, 592, "25%"],
```

```
            ["2009", 2342, 8958, 623, "27%"],
            ["2010", 2969, 11319, 617, "21%"],
            ["2011", 3528, 13573, 569, "16%"],
            ["2012", 3857, 14873, 586, "15%"],
            ["2013", 3725, 14316, 594, "16%"],
            ["2014", 3796, 14583, 583, "15%"],
            ["2015", 3698, 14285, 591, "16%"],
            ["2016", 3730, 14397, 595, "16%"],
            ["2017", 3932, 15174, 594, "15%"],
            ["2018", 3866, 14880, 593, "15%"],
            ["2019", 4054, 15493, 614, "15%"],
            ["2020", 4225, 16148, 770, "18%"]]

overall = pd.DataFrame(overall,
                       columns=["Year",
                                "Total Applicants",
                                "Total Applications",
                                "Places",
                                "Success Rate"])
```

[3]:
```
# Data Engineering: Applications per Student
overall["Applications per Student"] = overall["Total Applications"] /␣
 ↪overall["Total Applicants"]
overall["Applications per Student"] = overall["Applications per Student"].
 ↪apply(lambda x: round(x,2))

overall["Success (%)"] = (overall["Places"]/overall["Total Applicants"]*100)
overall["Success (%)"] = overall["Success (%)"].apply(lambda x: round(x,2))
overall
```

[3]:

	Year	Total Applicants	Total Applications	Places	Success Rate	\
0	2005	2125	7961	588	28%	
1	2006	2442	9152	554	23%	
2	2007	2346	8973	582	25%	
3	2008	2323	8566	592	25%	
4	2009	2342	8958	623	27%	
5	2010	2969	11319	617	21%	
6	2011	3528	13573	569	16%	
7	2012	3857	14873	586	15%	
8	2013	3725	14316	594	16%	
9	2014	3796	14583	583	15%	
10	2015	3698	14285	591	16%	
11	2016	3730	14397	595	16%	
12	2017	3932	15174	594	15%	
13	2018	3866	14880	593	15%	
14	2019	4054	15493	614	15%	
15	2020	4225	16148	770	18%	

```
    Applications per Student  Success (%)
0                       3.75         27.67
1                       3.75         22.69
2                       3.82         24.81
3                       3.69         25.48
4                       3.82         26.60
5                       3.81         20.78
6                       3.85         16.13
7                       3.86         15.19
8                       3.84         15.95
9                       3.84         15.36
10                      3.86         15.98
11                      3.86         15.95
12                      3.86         15.11
13                      3.85         15.34
14                      3.82         15.15
15                      3.82         18.22
```

Original files are in two different formats: - 2005 to 2012 - 2013 to 2020

[4]:
```
# We are going to create and append every dataframe to a list and the␣
↪concatenate all dataframes
empty =[]

for year in range(2005,2013):
    df = pd.read_excel("data/numbersplaces{}.xlsx".format(str(year)),
                       sheet_name = "{} Entry".format(str(year)),
                       skiprows=12,
                       engine="openpyxl")
    df.drop(columns=["Unnamed: 3","Unnamed: 4","Unnamed: 5","Unnamed: 6"],␣
↪inplace=True)
    df["Year"] = str(year)
    empty.append(df)
```

[5]:
```
for year in range(2013,2021):
    df = pd.read_excel("data/numbersplaces{}.xlsx".format(str(year)),
                           sheet_name = "{} Entry".format(str(year)),
                           skiprows=18,
                           engine="openpyxl")
    df = df[['Applications', 'Places', 'Course Centre']]
    df = df.dropna()
    df = df[df["Applications"] != "Applications"]
    df["Year"] = str(year)
    empty.append(df)
```

[6]:
```
applications = pd.concat(empty)
```

[7]:
```
applications["Type"] = applications.apply(
    lambda x: "Self-funded" if "self" in x["Course Centre"] else "Funded",␣
 ↪axis=1)

applications["Ratio University (%)"] = applications["Places"]*100/
 ↪applications["Applications"]
applications["Ratio University (%)"] = applications["Ratio University (%)"].
 ↪apply(lambda x: round(x,2))
applications["Places"] = applications["Places"].astype(float)
```

[8]:
```
applications.tail()
```

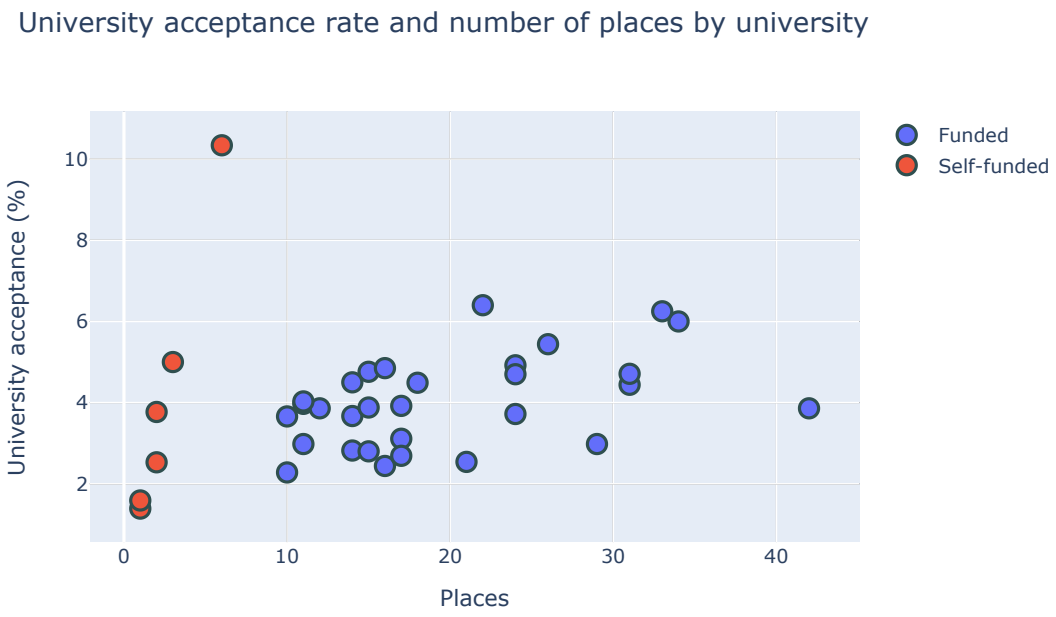
[8]:
```
    Applications  Places                Course Centre  Year         Type  \
41            90     1.0      Manchester - self-funded  2020  Self-funded
42            84     2.0       Newcastle - self-funded  2020  Self-funded
43            61     1.0        Plymouth - self-funded  2020  Self-funded
44            84     3.0  Royal Holloway - self-funded  2020  Self-funded
45            72     2.0       Sheffield - self-funded  2020  Self-funded

    Ratio University (%)
41                  1.11
42                  2.38
43                  1.64
44                  3.57
45                  2.78
```

1.0.1 Bubble plot: University acceptance rate and number of places by university

In this graph, we can compare university programmes by their acceptance rate and their number of places. The idea for this graph is to implement a dropdown function on the dashboard. This function will allow us to choose specific years interactively.

[9]:
```
df =  applications[applications["Year"] == "2017"]

data = []
for typ in df["Type"].unique():
    trace = df[df["Type"] == typ]
    scatter = go.Scatter(x = trace["Places"],
                         y = trace["Ratio University (%)"],
                         name = typ,
                         text = trace["Course Centre"],
                         mode = "markers",
                         marker=dict(size=12,
                                line=dict(width=2,
                                          color='DarkSlateGrey')
                         ))
    data.append(scatter)
```

```
layout = go.Layout(title="University acceptance rate and number of places by␣
 ↪university",
                   yaxis= dict(title="University acceptance (%)"),
                   xaxis=dict(title="Places"))
fig = go.Figure(data=data, layout=layout)
pyo.iplot(fig)
```

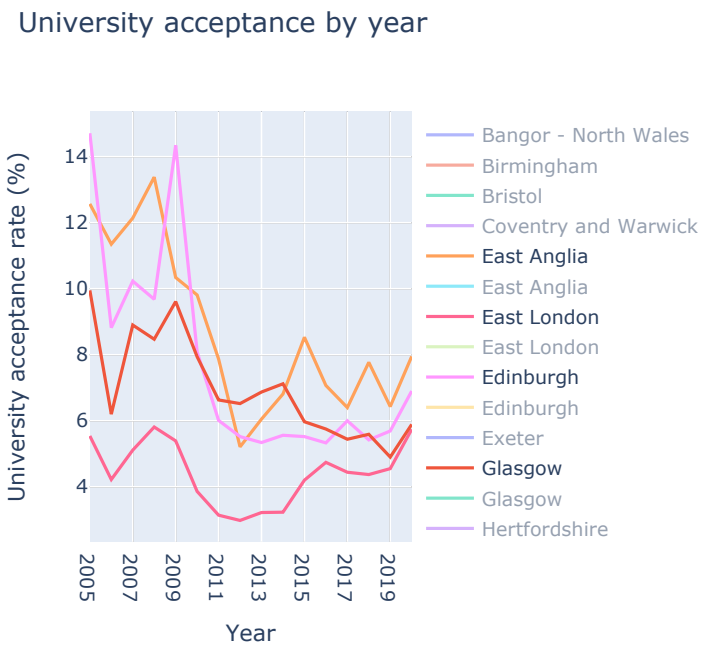


1.0.2 Interactive University acceptance graph

In this graph, we can compare university programmes by their acceptance rate by year interactively.

[10]:
```
# Use a for loop (or list comprehension to create traces for the data list)
data = []

for univ in applications["Course Centre"].unique():
    df2 = applications[applications["Course Centre"]==univ]
    if univ in ["East Anglia", "East London", "Glasgow", "Edinburgh"]:
        trace = go.Scatter(x=df2["Year"],
                   y=df2["Ratio University (%)"],
                   mode="lines",
                   name=univ)
        data.append(trace)
```

```
    trace = go.Scatter(x=df2["Year"],
                       y=df2["Ratio University (%)"],
                       mode="lines",
                       visible='legendonly',
                       name=univ)
    data.append(trace)

# Define the layout

layout = go.Layout(title="Exercise Motherfucker Scatterplot")

# Create a fig from data and layout, and plot the fig

layout = go.Layout(title="University acceptance by year",
                   yaxis= dict(title="University acceptance rate (%)"),
                   xaxis=dict(title="Year"))
fig = go.Figure(data=data, layout=layout)

pyo.iplot(fig)
```

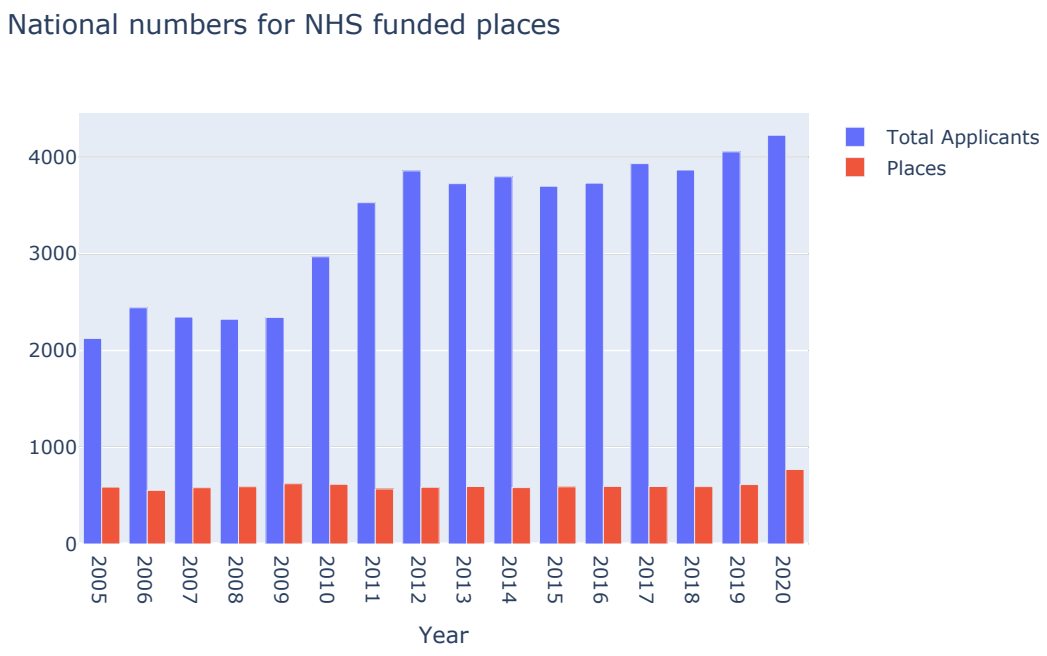


1.0.3 National numbers for NHS funded places

In this graph we can appreciate how the number of applicants have soared from 2009, while the number of NHS funded places have remained steady. However, in 2020 the NHS expanded the

number of places by 20%.

[11]:
```
df = overall

trace1 = go.Bar(x = df["Year"],
                y = df["Total Applicants"],
                name="Total Applicants")
trace2 = go.Bar(x = df["Year"],
                y = df["Places"],
                name="Places")



data = [trace1, trace2]

layout = go.Layout(title = "National numbers for NHS funded places",
                   barmode="group",
                   xaxis= dict(title="Year"))

fig = go.Figure(data=data, layout=layout)

pyo.iplot(fig)
```



[12]:
```
overall["Applicants Yearly growth"] = overall["Total Applicants"].pct_change()
```

[13]:
```
overall["Applications Yearly growth"] = overall["Total Applications"].
↪pct_change()
overall["Places Yearly growth"] = overall["Places"].pct_change()
```

[14]:
```
overall.head()
```

[14]:

	Year	Total Applicants	Total Applications	Places	Success Rate	\
0	2005	2125	7961	588	28%	
1	2006	2442	9152	554	23%	
2	2007	2346	8973	582	25%	
3	2008	2323	8566	592	25%	
4	2009	2342	8958	623	27%	

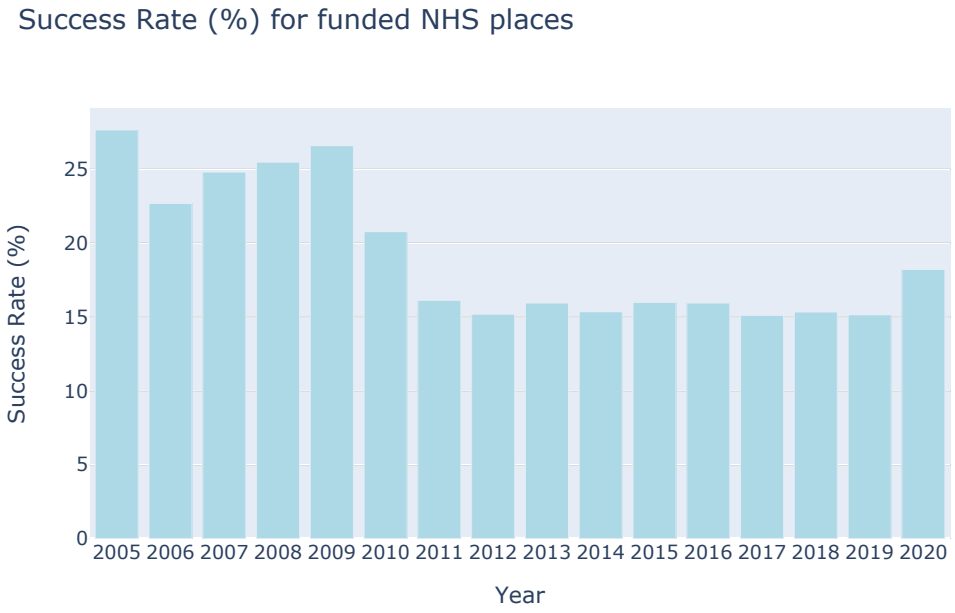
	Applications per Student	Success (%)	Applicants Yearly growth	\
0	3.75	27.67	NaN	
1	3.75	22.69	0.149176	
2	3.82	24.81	-0.039312	
3	3.69	25.48	-0.009804	
4	3.82	26.60	0.008179	

	Applications Yearly growth	Places Yearly growth
0	NaN	NaN
1	0.149604	-0.057823
2	-0.019559	0.050542
3	-0.045358	0.017182
4	0.045762	0.052365

2 Success Rate (%) for funded NHS places plot

[15]:
```
df = overall

trace1 = go.Bar(x = df["Year"],
                y = df["Success (%)"],
                name="Success (%)",
                marker=dict(color="lightblue"))


data = [trace1]

layout = go.Layout(title = "Success Rate (%) for funded NHS places",
                   barmode="group",
                   yaxis= dict(title="Success Rate (%)"),
                   xaxis= dict(title="Year"))

fig = go.Figure(data=data, layout=layout)
```

```
pyo.iplot(fig)
```



From 2005 until 2009, the success rate was around 25%, and the number of applicants was around 2250, remaining both relatively stable. From 2010 until now, the number of applicants has soared, reaching 4225 applicants by 2020. Simultaneously, the number of NHS places remained steady, averaging a success rate of 15% over these years

In 2020, the NHS increased the number of funded places by 20%, incrementing the overall success rate of the applicants to 18.22%.

2.0.1 Preliminary conclusions:

It seems that the first conclusion is to apply to four universities. Most candidates apply to four universities since the number of applications per student ratio was 3.82.

The second conclusion is that some universities are less competitive than others, and this analysis allows us to identify these universities. For now, we know some of these are: - East Anglia - East London - Glasgow - Edinburg

In the future, I plan to provide a historical ranking of the difficulty to enter each university, showing how much variability there is from one year to another.