Nurse Staffing Recommendations

Matthew Bain

2024-03-22

abstract I investigate nurse staffing data to provide informed recommendations to a medical staffing organization.

Table of contents

[…]

Imports

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from great_tables import GT
from pandas.plotting import scatter_matrix

from src.stylesheet import customize_plots
from src.inspection import make_df, display
```

The dataset

We begin by exploring the data to get to know the features and patterns on which we will base our analysis.

```python
if 'data' not in locals():
    data = pd.read_csv(
        "../data/raw/PBJ_Daily_Nurse_Staffing_Q1_2024.zip",
        encoding='ISO-8859-1'
    )
else:
    print("data loaded.")
```

```
data.sample(5)
```

	PROVNUM	PROVNAME	CITY	STATE	COUNTY_NAME	COUNTY_FIPS	CY_Qtr	WorkDate	MDScensus	Hrs_RNDON	…	Hrs_LPN_ctr	Hrs_CNA	Hrs_CNA_emp	Hrs_CNA_ctr	Hrs_NAtrn	Hrs_NAtrn_emp	Hrs_NAtrn_ctr	Hrs_MedAide	Hrs_MedAide_emp	Hrs_MedAide_ctr
134826	075198	VILLAGE GREEN OF BRISTOL REHAB & HEALTH CENTER	FORESTVILLE	CT	Hartford	3	2024Q1	20240228	84	0.0	…	0.00	129.16	129.16	0.00	11.63	11.63	0.0	0.00	0.00	0.0
314429	146128	PLYMOUTH PLACE	LA GRANGE PARK	IL	Cook	31	2024Q1	20240127	52	8.0	…	0.00	183.59	183.59	0.00	0.00	0.00	0.0	0.00	0.00	0.0
649459	265607	CRYSTAL CREEK HEALTH AND REHABILITATION CENTER	FLORISSANT	MO	St. Louis	189	2024Q1	20240320	149	0.0	…	0.00	149.00	149.00	0.00	0.00	0.00	0.0	50.25	50.25	0.0
992920	395640	JOHN J KANE REGIONAL CENTER-MC	MCKEESPORT	PA	Allegheny	3	2024Q1	20240120	198	0.0	…	100.25	450.75	224.50	226.25	0.00	0.00	0.0	0.00	0.00	0.0
661170	265793	LA PLATA NURSING HOME	LA PLATA	MO	Macon	121	2024Q1	20240223	58	0.0	…	0.00	83.33	83.33	0.00	11.75	11.75	0.0	14.38	14.38	0.0

```
# TODO: pivot on day

data_pivoted = data.pivot_table(
    index="STATE",
    columns="WorkDate",
    values="Hrs_RN",
    aggfunc='mean'
)

# Resetting the index for easier column access
# data_pivoted.reset_index(inplace=True)
data_pivoted
```

Week Date STATE	[illegible]
AK	[illegible]
AL	[illegible]
AR	[illegible]
AZ	[illegible]
CA	[illegible]
CO	[illegible]
CT	[illegible]
DC	[illegible]
DE	[illegible]
FL	[illegible]
GA	[illegible]
HI	[illegible]
IA	[illegible]
ID	[illegible]
IL	[illegible]
IN	[illegible]
KS	[illegible]
KY	[illegible]
LA	[illegible]
MA	[illegible]
MD	[illegible]
ME	[illegible]
MI	[illegible]
MN	[illegible]
MO	[illegible]
MS	[illegible]
MT	[illegible]
NC	[illegible]
ND	[illegible]
[illegible]	[illegible]

```
data_pivoted.iloc[:, 1:]
```

Week Date	[illegible]
STATE	
AK	[illegible]
AL	[illegible]
AR	[illegible]
AZ	[illegible]
CA	[illegible]
CO	[illegible]
CT	[illegible]
DC	[illegible]
DE	[illegible]
FL	[illegible]
GA	[illegible]
HI	[illegible]
IA	[illegible]
ID	[illegible]
IL	[illegible]
IN	[illegible]
KS	[illegible]
KY	[illegible]
LA	[illegible]
MA	[illegible]
MD	[illegible]
ME	[illegible]
MI	[illegible]
MN	[illegible]
MO	[illegible]
MS	[illegible]
MT	[illegible]
NC	[illegible]
ND	[illegible]
[illegible]	[illegible]

```
# (
#     GT(data_pivoted, rowname_col="STATE")
#     .fmt_nanoplot(
#         columns=data_pivoted.columns[1:],
#         reference_line="mean",
#         reference_area=["min", "q1"]
#     )
#     .fmt_nanoplot(
#         columns=data_pivoted.columns[1:],
#         plot_type="bar",
#         reference_line="max",
#         reference_area=["max", "median"]
#     )
# )
```

```
data.describe().round(1)
# display(Markdown(data.describe().to_markdown()))
```

	COUNTY_FIPS	WorkDate	MDScensus	Hrs_RNDON	Hrs_RNDON_emp	Hrs_RNDON_ctr	Hrs_RNadmin	Hrs_RNadmin_emp	Hrs_RNadmin_ctr	Hrs_RN	…	Hrs_LPN_ctr	Hrs_CNA	Hrs_CNA_emp	Hrs_CNA_ctr	Hrs_NAtrn	Hrs_NAtrn_emp	Hrs_NAtrn_ctr	Hrs_MedAide	Hrs_MedAide_emp	Hrs_MedAide_ctr
count	1330966.0	1330966.0	1330966.0	1330966.0	1330966.0	1330966.0	1330966.0	1330966.0	1330966.0	1330966.0	…	1330966.0	1330966.0	1330966.0	1330966.0	1330966.0	1330966.0	1330966.0	1330966.0	1330966.0	1330966.0
mean	92.0	20240215.4	83.7	5.2	5.1	0.1	10.3	10.0	0.2	34.4	…	6.5	171.2	158.2	13.0	4.2	4.2	0.1	8.5	8.3	0.2
std	99.2	83.0	49.1	4.5	4.5	0.9	14.9	14.6	1.8	34.7	…	16.2	113.7	106.3	32.6	13.1	12.7	2.1	17.6	17.2	2.2
min	1.0	20240101.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
25%	31.0	20240125.0	51.0	0.0	0.0	0.0	0.0	0.0	0.0	13.0	…	0.0	97.0	88.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
50%	69.0	20240215.0	76.0	8.0	8.0	0.0	7.5	7.4	0.0	25.6	…	0.0	148.1	136.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0
75%	127.0	20240309.0	104.0	8.0	8.0	0.0	16.0	16.0	0.0	44.8	…	5.8	217.0	203.1	11.0	0.0	0.0	0.0	11.2	10.8	0.0
max	840.0	20240331.0	713.0	327.8	327.8	42.0	266.2	266.2	92.5	908.6	…	454.0	1857.1	1573.6	494.3	452.0	279.0	280.5	395.6	395.6	128.9

```
attributes = ["Hrs_RN", "Hrs_LPN_ctr", "Hrs_CNA", "Hrs_NAtrn", "Hrs_MedAide"]
n = len(attributes)

fig, axs = plt.subplots(n, n, figsize=(8, 8))
scatter_matrix(
    data[attributes].sample(200),
    ax=axs, alpha=.7,
    hist_kwds=dict(bins=15, linewidth=0)
)
fig.align_ylabels(axs[:, 0])
fig.align_xlabels(axs[-1, :])
```

```
for ax in axs.flatten():
    ax.tick_params(axis='both', which='both', length=3.5)

# save_fig("scatter_matrix_plot")

plt.show()
```

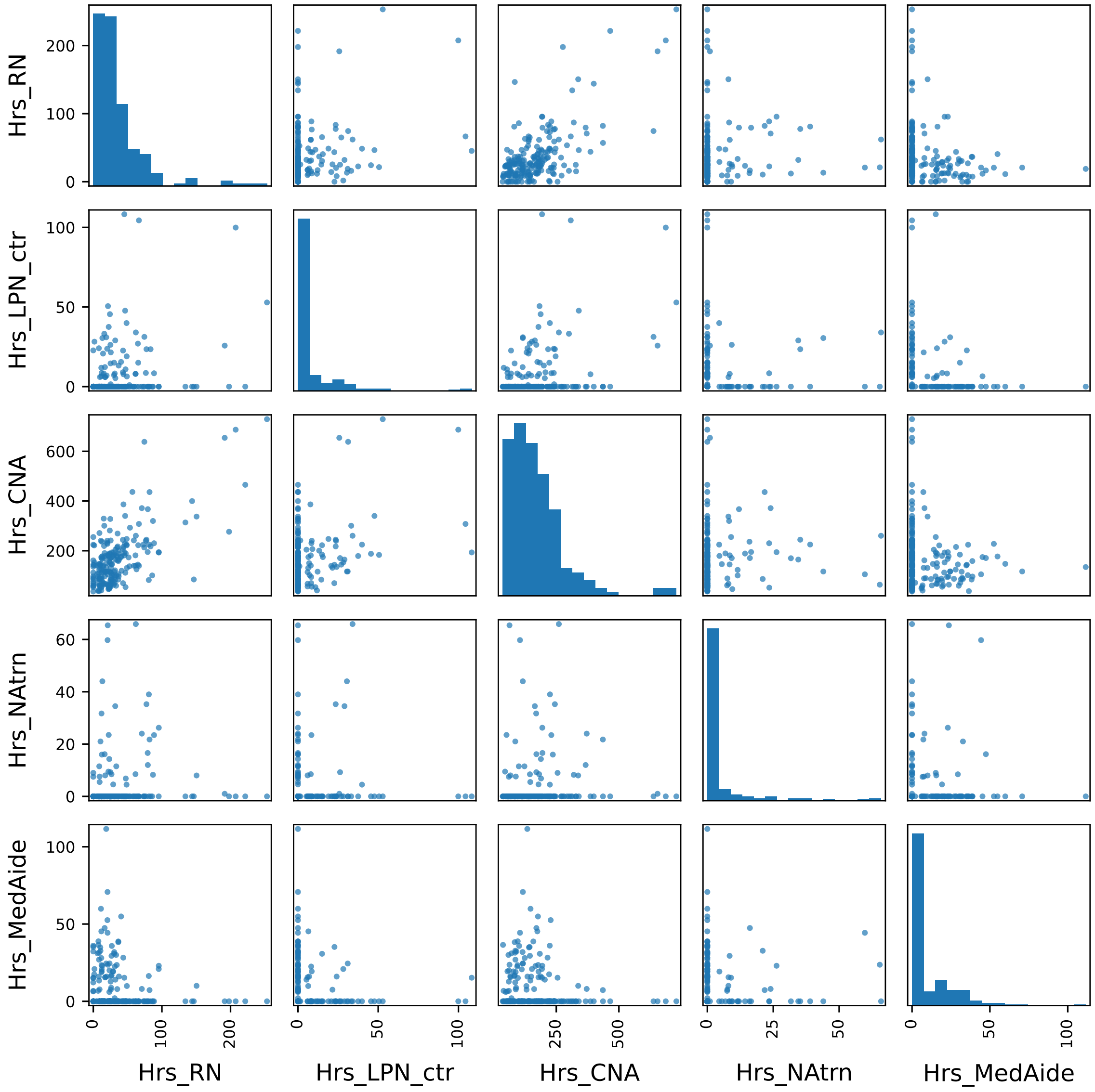

Figure 1: Scatter matrix of nursing worker working hours

```
import pandas as pd
df = pd.DataFrame({'name': ['arizona', '', 'berlin', 'london']})
```

```
from geopy.geocoders import Nominatim
geolocator = Nominatim(user_agent="geo_clipboard")

from geopy.extra.rate_limiter import RateLimiter
geocode = RateLimiter(geolocator.geocode, min_delay_seconds=1)
df['location'] = df['name'].apply(geocode)

df['point'] = df['location'].apply(lambda loc: tuple(loc.point) if loc else None)
```

```
df
```

	name	location	point
0	arizona	(Arizona, United States, (34.395342, −111.7632…	(34.395342, −111.763275, 0.0)
1		None	None
2	berlin	(Berlin, Deutschland, (52.510885, 13.3989367))	(52.510885, 13.3989367, 0.0)
3	london	(London, Greater London, England, United Kingd…	(51.4893335, −0.14405508452768728, 0.0)

```
from great_tables import GT

df = data.loc[150000:, [
    "STATE",
    "COUNTY_NAME", "COUNTY_FIPS",
    "CITY",
    "PROVNAME", "PROVNUM",
    # "MDScensus"
]].value_counts().reset_index()
GT(df.head(n=10))
```

STATE	COUNTY_NAME	COUNTY_FIPS	CITY	PROVNAME	PROVNUM	count
CA	Alameda	1	ALAMEDA	ALAMEDA HEALTHCARE & WELLNESS CENTER	555486	91
OH	Mahoning	99	AUSTINTOWN	AVENTURA AT HUMILITY HOUSE	366186	91
OH	Lucas	95	TOLEDO	OHIO LIVING SWAN CREEK	365996	91
OH	Lucas	95	TOLEDO	OTTERBEIN SUNSET HOUSE	366148	91
OH	Lucas	95	TOLEDO	PARK TERRACE NURSING AND REHABILITATION CENTER	365339	91
OH	Lucas	95	TOLEDO	POINT PLACE HEALTHCARE AND REHABILITATION CENTER	366039	91
OH	Lucas	95	WATERVILLE	ASTORIA PLACE OF WATERVILLE	365747	91
OH	Lucas Madison	95 97	WHITEHOUSE LONDON [illegible]	[illegible]	365[illegible]	91

Some GT examples

```
from typing import Any
from IPython.display import display as ipy_display, HTML
import numpy as np

def display2(
    *args,
    globs: dict[str, Any] | None = None,
    bold: bool = True,
    width: str = "400px"  # Fixed width for each block
) -> None:
    """
    Display an informative representation of multiple objects side-by-side in
Jupyter.

    Parameters
    ----------
    *args : tuple
        Tuple of expressions to evaluate and display.
    globs : dict[str, Any], default=None
        Global namespace, to give eval() access to nonlocals passed by name.
    bold : bool, default=True
        Option to enable/disable string styling.
    width : str, default="400px"
        Fixed width for each displayed block in the Jupyter notebook.

    Warnings
    --------
    This function uses `eval()` to render expressions it receives
    as strings. Access to variables in the global namespace is controlled
    by `globs`. Take care to only pass trusted expressions to the function.
    """

    if globs is None:
        globs = {}

    outputs = []
    for arg in args:
        name = f"<b>{arg}</b>" if bold else arg
        value = np.round(eval(arg, globs), 2)
        shape = np.shape(value)
                   content = f"<div style='width:{width}; padding:10px;
float:left;'><pre>{name}\n--- {repr(shape)} ---\n{repr(value)}</pre></div>"
        outputs.append(content)

    # Clearfix for layout
    clearfix = "<div style='clear: both;'></div>"
    # Display the HTML content in Jupyter
    html_output = ''.join(outputs) + clearfix
    ipy_display(HTML(html_output))

    return None
```

```
A = np.array([[1, 3], [2, 4]])
x = np.array([[0, 1]])

display2(
    "A", "x.T", "np.dot(A, x.T)", globs=globals(), bold=True, width="100px"
)
```

```
<IPython.core.display.HTML object>
```

```
display2(
    "data['STATE'].value_counts()",
    "data['COUNTY_NAME'].value_counts()",
    "data['CITY'].value_counts()",
    "data['PROVNAME'].value_counts()",
    "data['MDScensus'].value_counts()",
    width="340px",
    globs=globals()
)
```

```
<IPython.core.display.HTML object>
```

```
data[["CY_Qtr", "WorkDate", "MDScensus"]]
```

	CY_Qtr	WorkDate	MDScensus
0	2024Q1	20240101	50
1	2024Q1	20240102	49
2	2024Q1	20240103	49
3	2024Q1	20240104	50
4	2024Q1	20240105	51
…	…	…	…
1330961	2024Q1	20240327	81
1330962	2024Q1	20240328	83
1330963	2024Q1	20240329	85
1330964	2024Q1	20240330	82
1330965	2024Q1	20240331	82

SQL

Bibliography