blood_pressure_analysis

February 19, 2026

1 Blood Pressure Data Analysis

This analysis evaluates blood pressure readings to determine:

1. How many patients have elevated blood pressure
2. A visualization of a single patient with uncontrolled blood pressure
3. A provider-level comparison of elevated blood pressure prevalence

[17]:

```
import pandas as pd
```

[18]:

```
import pandas as pd

df = pd.read_csv("blood_pressure_data.csv")
df.head()
```

[18]:

```
         Date         Time Patient First Name Patient Last Name        PCP  \
0   6/23/2021  11:41:00 AM                  A                 A  Dr. Clark
1  10/26/2021   1:45:00 PM                  A                 A  Dr. Clark
2   9/11/2021   1:05:00 PM                  C                 A  Dr. Clark
3   9/11/2021   1:07:00 PM                  C                 A  Dr. Clark
4   1/15/2021  10:33:00 AM                  G                 A   Dr. Fish

  Clinic Visit Provider               Clinic  Systolic BP reading  \
0             Dr. Clark                  GIM                  149
1             Dr. Basit  Parkland Cardiology                  141
2             Dr. Basit      UTSW Cardiology                  110
3             Dr. Basit      UTSW Cardiology                  110
4             Dr. Basit  Parkland Cardiology                  168

   Diastolc BP reading
0                   89
1                   64
2                   67
3                   63
4                  103
```

[19]:

```
# misspelled column
df = df.rename(columns={"Diastolc BP reading": "Diastolic BP reading"})
```

[20]:

```
df.columns
```

[20]:

```
Index(['Date', 'Time', 'Patient First Name', 'Patient Last Name', 'PCP',
       'Clinic Visit Provider', 'Clinic', 'Systolic BP reading',
       'Diastolic BP reading'],
      dtype='str')
```

[21]:

```
df.shape
```

[21]:

```
(154, 9)
```

[22]:

```
df.isna().sum()
```

[22]:

```
Date                     0
Time                     0
Patient First Name       0
Patient Last Name        0
PCP                      0
Clinic Visit Provider    0
Clinic                   0
Systolic BP reading      0
Diastolic BP reading     0
dtype: int64
```

[23]:

```
# create a pt ID
df["Patient_ID"] = df["Patient First Name"] + "_" + df["Patient Last Name"]

df["Patient_ID"].head()
```

[23]:

```
0    A_A
1    A_A
2    C_A
3    C_A
4    G_A
Name: Patient_ID, dtype: str
```

[24]:

```
# categorize blood pressure and define normal, high or low
def bp_category(sbp, dbp):
    if (sbp < 90) or (dbp < 60):
        return "Low"
    if (sbp >= 130) or (dbp >= 80):
        return "Elevated"
    return "Normal"

df["BP_Category"] = df.apply(
    lambda row: bp_category(row["Systolic BP reading"], row["Diastolic BP␣
↪reading"]),
    axis=1
)

df["BP_Category"].value_counts()
```

[24]:

```
BP_Category
Elevated    101
Low          40
Normal       13
Name: count, dtype: int64
```

[25]:

```
# Identify elevated readings
df["is_elevated"] = df["BP_Category"] == "Elevated"

# Group by patient and check if they EVER had an elevated reading
patient_elevated = (
    df.groupby("Patient_ID")["is_elevated"]
      .any()
)

total_patients = df["Patient_ID"].nunique()
elevated_patients = patient_elevated.sum()

print("Total patients:", total_patients)
print("Patients with  1 elevated reading:", elevated_patients)
print("Percent elevated:", round(100 * elevated_patients / total_patients, 1),␣
↪"%")
```

```
Total patients: 74
Patients with  1 elevated reading: 58
Percent elevated: 78.4 %
```

1.1 Part 1: How Many Patients Have Elevated Blood Pressure?

Blood pressure readings were categorized as Low, Normal, or Elevated using predefined systolic and diastolic thresholds.

A patient was considered to have elevated blood pressure if they had at least one reading categorized as “Elevated.”

After grouping readings by patient and checking whether each patient ever had an elevated reading:

- Total unique patients: 74
- Patients with 1 elevated reading: 58
- Percent of patients with elevated blood pressure: 78.4%

This analysis moves from reading-level data to patient-level classification by using a group-by operation and checking whether any elevated readings occurred for each patient.

[26]:

```
# create a visualization for elevated BP, interested in pt with uncontrolled BP
# Calculate percent elevated readings per patient
patient_summary = (
    df.groupby("Patient_ID")["is_elevated"]
      .mean()
      .reset_index()
)

patient_summary.columns = ["Patient_ID", "Percent_Elevated"]

patient_summary.sort_values("Percent_Elevated", ascending=False).head()
```

[26]:

```
  Patient_ID  Percent_Elevated
0        A_A               1.0
3        A_H               1.0
8        B_D               1.0
4        A_I               1.0
5        A_J               1.0
```

[27]:

```
example_patient = patient_summary.sort_values(
    "Percent_Elevated",
    ascending=False
)["Patient_ID"].iloc[0]

example_patient
```

[27]:

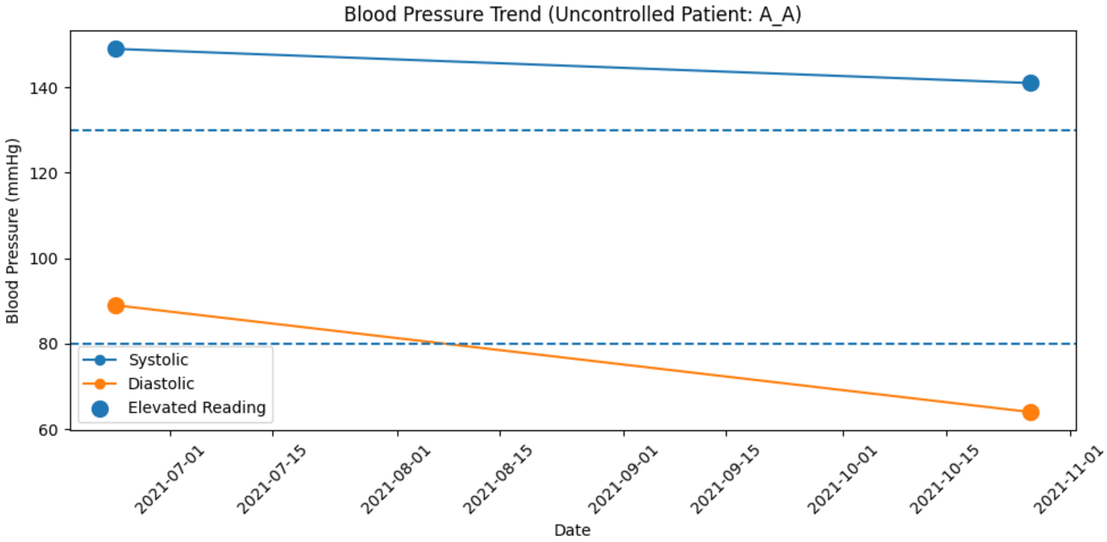
```
'A_A'
```

[28]:

```
patient_df = df[df["Patient_ID"] == example_patient].copy()
```

[29]:

```
# visualization single pt with elevated BP that is more uncontrolled
import matplotlib.pyplot as plt
import pandas as pd

# Ensure DateTime column exists
patient_df["DateTime"] = pd.to_datetime(
    patient_df["Date"] + " " + patient_df["Time"]
)

patient_df = patient_df.sort_values("DateTime")

plt.figure(figsize=(10,5))

# Plot systolic and diastolic trends
plt.plot(
    patient_df["DateTime"],
    patient_df["Systolic BP reading"],
    marker="o",
    label="Systolic"
)

plt.plot(
    patient_df["DateTime"],
    patient_df["Diastolic BP reading"],
    marker="o",
    label="Diastolic"
)

# Highlight elevated readings
elevated_points = patient_df[patient_df["BP_Category"] == "Elevated"]

plt.scatter(
    elevated_points["DateTime"],
    elevated_points["Systolic BP reading"],
    s=100,
    zorder=5,
    label="Elevated Reading"
)

plt.scatter(
    elevated_points["DateTime"],
    elevated_points["Diastolic BP reading"],
    s=100,
    zorder=5
)

# Optional reference lines for interpretation
plt.axhline(130, linestyle="--")
plt.axhline(80, linestyle="--")

plt.title(f"Blood Pressure Trend (Uncontrolled Patient: {example_patient})")
plt.xlabel("Date")
plt.ylabel("Blood Pressure (mmHg)")
plt.xticks(rotation=45)
plt.legend()
plt.tight_layout()
plt.show()
```



1.2 Part 2: Single Patient Visualization

A patient with the highest proportion of elevated readings (100%) was selected to represent uncontrolled blood pressure.

A time-series line chart was used to display systolic and diastolic blood pressure over time. Elevated readings were highlighted and clinical reference thresholds (130 mmHg systolic and 80 mmHg diastolic) were included for interpretation.

Strengths: - Displays trends over time - Clearly identifies persistent elevation - Clinically intuitive format

Weaknesses: - Does not account for medication changes - Limited number of observations - Does not assess long-term control patterns

[30]:

```
# Determine if each patient had ANY elevated reading with each provider
provider_patient = (
    df.groupby(["Clinic Visit Provider", "Patient_ID"])["is_elevated"]
      .any()
      .reset_index()
)

provider_patient.head()
```

[30]:

```
  Clinic Visit Provider Patient_ID  is_elevated
0             Dr. Basit        A_A         True
1             Dr. Basit        A_B        False
2             Dr. Basit        B_B         True
3             Dr. Basit        B_D         True
4             Dr. Basit        B_H        False
```

[31]:

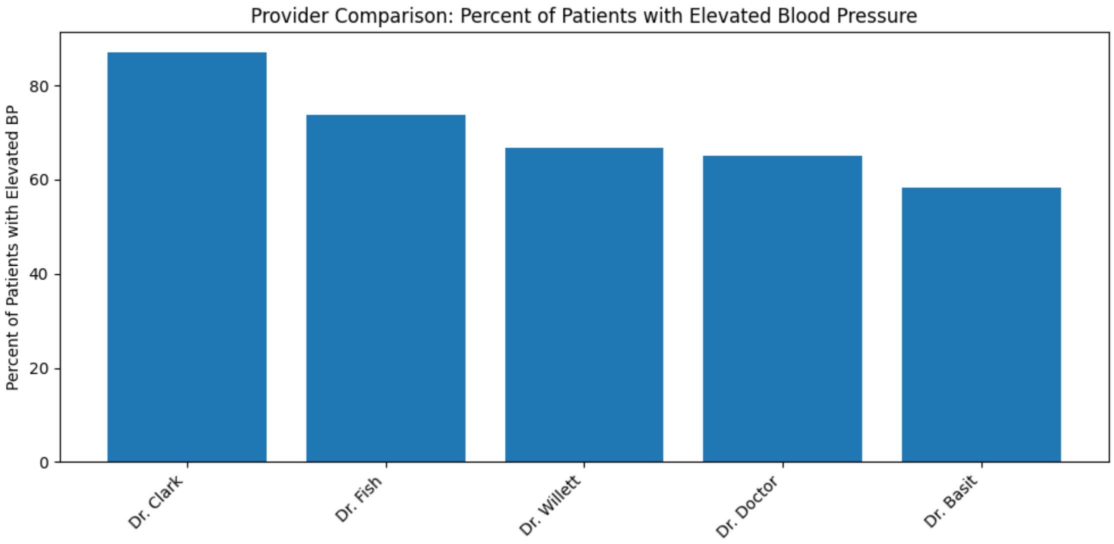
```
# percent of elevated pts per MD
provider_summary = (
    provider_patient.groupby("Clinic Visit Provider")["is_elevated"]
    .mean()
    .reset_index()
)

provider_summary.columns = ["Provider", "Percent_Elevated_Patients"]

provider_summary["Percent_Elevated_Patients"] *= 100

provider_summary.sort_values("Percent_Elevated_Patients", ascending=False)
```

[31]:

```
      Provider  Percent_Elevated_Patients
1    Dr. Clark                  86.956522
3     Dr. Fish                  73.684211
4  Dr. Willett                  66.666667
2   Dr. Doctor                  65.000000
0    Dr. Basit                  58.333333
```

[32]:

```
# visualization for provider
import matplotlib.pyplot as plt

provider_summary = provider_summary.sort_values(
    "Percent_Elevated_Patients",
    ascending=False
)

plt.figure(figsize=(10,5))

plt.bar(
    provider_summary["Provider"],
    provider_summary["Percent_Elevated_Patients"]
)

plt.ylabel("Percent of Patients with Elevated BP")
plt.title("Provider Comparison: Percent of Patients with Elevated Blood␣
 ↪Pressure")
plt.xticks(rotation=45, ha="right")

plt.tight_layout()
plt.show()
```



1.3 Part 3: Provider-Level Visualization

To evaluate how each provider is managing blood pressure within their patient population, I calculated the percentage of patients per provider who had at least one elevated blood pressure reading.

For each provider–patient combination, it was determined whether the patient ever had an elevated reading, and then the proportion of patients with elevated blood pressure was calculated for each provider.

Strengths: - Provides population-level comparison - Easy visual interpretation - Aligns with patient-level classification from Part 1

Weaknesses: - Does not adjust for patient risk or severity - Does not account for improvement over time - Differences may reflect patient mix rather than provider performance

1.4 Overall Interpretation

A substantial proportion of patients in this dataset had elevated blood pressure (78.4%). Provider-level analysis shows variation in the proportion of patients with elevated readings, though these differences may reflect patient population characteristics or service line differences rather than provider performance.

The single-patient visualization demonstrates how persistent elevation can be identified through time-series analysis.