In [1]:
```
# Standard library
import numpy as np
import pandas as pd
import random
import os

# Visualization libraries
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
import plotly.io as pio
from IPython.display import Image

# Custom modules
from empirical_null import empirical_null_provider_failures, bootstrap_stabi
from hierarchical_bayes import fit_hierarchical_logistic_model, plot_bayesia
from survival_analysis import bayesian_cox_model, plot_bayesian_cox_effects
from utils import calculate_vif, generate_later_timestamp_column, plot_outcc

import warnings
warnings.filterwarnings('ignore')
random.seed(54)  # For reproducibility

output_dir = "plots"
os.makedirs(output_dir, exist_ok=True)
```

Clover Health Technical

Overall Goal: Using data provided, qualify/quantify doctor's efficacy with respect to a certain medical condition to provide data-driven guidance on whether or not an insurance company considering adding physicians should have concerns about any/all individual physicians.

1. If you were to only consider the provider's effectiveness with regard to this particular treatment, would you recommend reaching out to any of these doctors in particular? If so, which ones, and why?

Strategy:

- Exploratory data analysis: understand if there is any bias or trends that may be affecting the data, i.e., do any physicians only treat certain types of patients in terms of covariates? Is there a time bias to patient outcomes? Are any other covariates highly associated with outcome? How many patients does each physician see? etc.
- Evaluate effect of physician on outcome
  - Assumptions:

    - NaN in outcome is a positive result (lack of health failure)
    - Size of data provided is equivalent to size of population (solutions are not optimized for scale)
- Method 1:
    - Use a hierarchical Bayesian logistic regression to determine the probability that any given provider has an increased likelihood of a failure while accounting for covariates
- Method 2:
    - Use a covariate-adjusted empirical null to determine whether any providers have significantly different than expected failure rates while accounting for covariates

Exploratory Data Analysis

In [2]:
```
data = pd.read_csv("data/Product_Data.csv")
```

In [3]:
```
# number of providers and number of patients
print(f"number of unique provider ids : {data.servicing_provider_id.nunique(
print(f"number of unique patient ids : {data.member_id.nunique()}")

# we also have servicing name, how does this map to id
print(data[["servicing_provider_id", "servicing_provider_name"]].drop_duplic
# looks like there is a one to one mapping for each servicing provider name

# lets understand patient results

# are there any missing values
print(data.outcome.isna().value_counts())
# yes between 1/2-1/3 are missing outcomes

# what are the nonmissing outcomes
print(data.outcome.value_counts())
print(f"outcomes are {data.outcome.unique()}")
# so we come to our first assumption; Na's in outcome are successes (only al
# in which case we can either take a comparartive approach or a survival app

# is there only 2 outcome per patient?
print(data.groupby("member_id").outcome.nunique().value_counts() )
print(f"number of patients with multiple outcomes: {data.groupby('member_id'

# how many visits per patient
print(data.groupby("member_id").event_id.nunique().value_counts())
print(f"number of patients with multiple visits: {data.groupby('member_id').
# okay so we have a single outcome per patient, and a single visit per patie

print(data.groupby("member_id").servicing_provider_id.nunique().value_counts
print(f"number of patients with multiple providers: {data.groupby('member_ic
# all patients only see 1 provider

# lets just check for NA's
```

```
data.isna().value_counts()
# outcome is the only column with NA's
```

```
number of unique provider ids : 96
number of unique patient ids : 4247
count
1    96
Name: count, dtype: int64
outcome
False    2413
True     1834
Name: count, dtype: int64
outcome
failure    2413
Name: count, dtype: int64
outcomes are ['failure' nan]
outcome
1    2413
0    1834
Name: count, dtype: int64
number of patients with multiple outcomes: 0
event_id
1    4247
Name: count, dtype: int64
number of patients with multiple visits: 0
servicing_provider_id
1    4247
Name: count, dtype: int64
number of patients with multiple providers: 0
```

Out[3]:

```
event_id  servicing_provider_id  servicing_provider_name  treatment_date  m
ember_id  member_age  member_sex  health_risk_assesment  outcome
False     False                  False                   False          F
alse      False       False       False                  False      2413

True       1834
Name: count, dtype: int64
```

In [4]:

```
# what about how many patients each provider sees
patientsxprovider = data.groupby("servicing_provider_id").member_id.nunique(
print(f"providers have on average {patientsxprovider.mean():.2f} patients, w
fig = px.histogram(patientsxprovider, title="Patients per Provider")
fig.add_vline(x=patientsxprovider.mean(), line_color="red", line_dash="dash"
fig.show()

fig_path = os.path.join(output_dir, "patientsxproviders.png")
pio.write_image(fig, fig_path)
```

```
providers have on average 44.24 patients, with a std of 6.36
```

In [5]:

```
# what about how many patients each provider sees
outcomesxprovider = data.groupby("servicing_provider_id").outcome.count()
# lets look at this with respect to proportion of patients that fail
outcomes_prop_xprovider = outcomesxprovider.values / patientsxprovider.value
fig = px.histogram(outcomes_prop_xprovider)
fig.show()
# proportion of patients that fail is normally distributed, maybe slightly s
```

```
fig_path = os.path.join(output_dir, "outcomesxprovider.png")
pio.write_image(fig, fig_path)
```

In [6]:
```
# health risk assessment by provider
hrasxprovider = data.groupby("servicing_provider_id").health_risk_assesment.
print(hrasxprovider.value_counts())
print("7 providers see patients with only 1 health risk assessment score, mo
```

```
health_risk_assesment
9     62
10    19
8     14
7      1
Name: count, dtype: int64
7 providers see patients with only 1 health risk assessment score, most prov
iders see patients with several health risk assessment scores
```

In [7]:
```
# ASSUMPTION 1 : NA in outcome is a success
data_outcome_compare = data.copy()
data_outcome_compare["outcome"] = data.outcome.fillna("pass")
```

In [8]:
```
# plot by time
data_by_time = data.groupby(["treatment_date", "outcome"]).member_id.nunique
data_by_time["treatment_date"] = pd.to_datetime(data_by_time["treatment_date

fig = px.line(data_by_time.sort_values("treatment_date"), x="treatment_date"
fig.show()
# doesnt appear to be a strong trend over time
fig_path = os.path.join(output_dir, "outcomextime.png")
pio.write_image(fig, fig_path)
```

In [9]:
```
# lets check member age too
data_outcome_compare["member_age"].isna().value_counts()
# no missing values
data_outcome_compare_count = data_outcome_compare.groupby(["member_age", "he
fig = px.scatter(x="health_risk_assesment", y="member_age", size="member_id"
age_risk_corr = data_outcome_compare[["health_risk_assesment", "member_age"]
fig.add_annotation(
    xref="paper", yref="paper", x=0.05, y=0.95,
    text=f"Correlation: {age_risk_corr:.2f}",
    showarrow=False,
    font=dict(size=14, color="black"),
    bgcolor="white"
)
fig.show()
fig_path = os.path.join(output_dir, "agexoutcomexhra.png")
pio.write_image(fig, fig_path)
# doesnt appear to be a strong relationship between age and outcome, but we
```

In [10]:
```
cat_plot = data_outcome_compare[["member_sex", "health_risk_assesment", "out
cat_plot["member_sex"] = cat_plot["member_sex"].replace({0:"F", 1:"M"})
fig = px.bar(cat_plot, x="health_risk_assesment", y="count", color="outcome"
fig.show()
# Females do seem to have more positive outcomes at higher health risk asses
```

```
fig_path = os.path.join(output_dir, "sexxoutcomexhra.png")
pio.write_image(fig, fig_path)
```

Looks like there is a moderate global correlation between age and health risk assesment and we know from our information that age and sex are incorporated into this measure. Since we plan to use bayesian methods we should probably assess for colinearity

In [11]:
```
vif_df = calculate_vif(data_outcome_compare, ["member_age", "health_risk_ass
fig = px.bar(vif_df, x="feature", y="VIF", title="Variance Inflation Factor
fig.show()
high_vif_features = vif_df.query("VIF > 5")
if not high_vif_features.empty:
    print(f"Warning: The following features have VIF > 5 and indicate multic
    print("One of these will be dropped for downstream analysis to reduce co

fig_path = os.path.join(output_dir, "vif.png")
pio.write_image(fig, fig_path)
```

```
Warning: The following features have VIF > 5 and indicate multicollinearity:
['member_age', 'health_risk_assesment']
One of these will be dropped for downstream analysis to reduce collinearity.
```

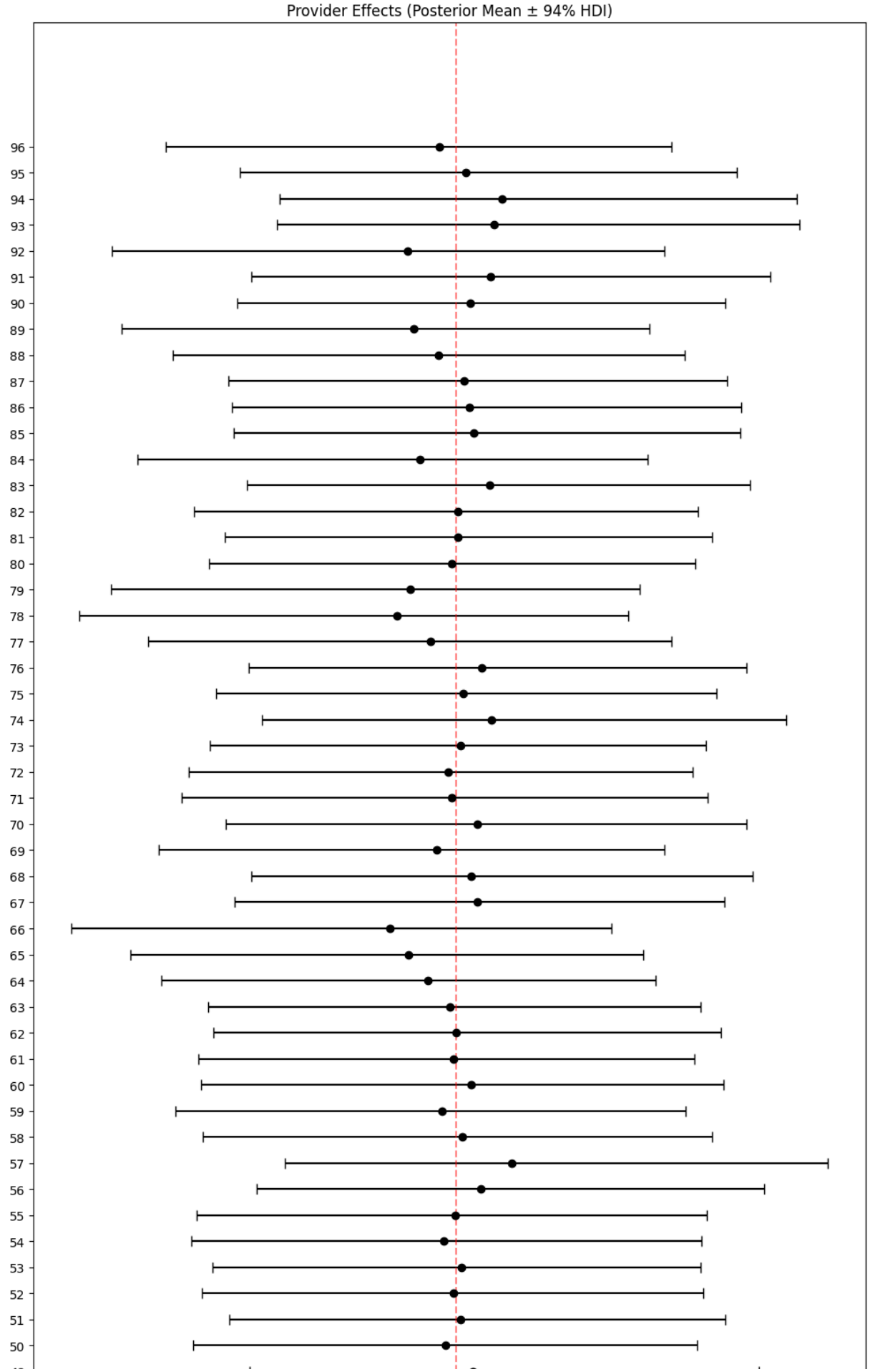
Method 1 : Hierarchical Bayesian Logistic Regression

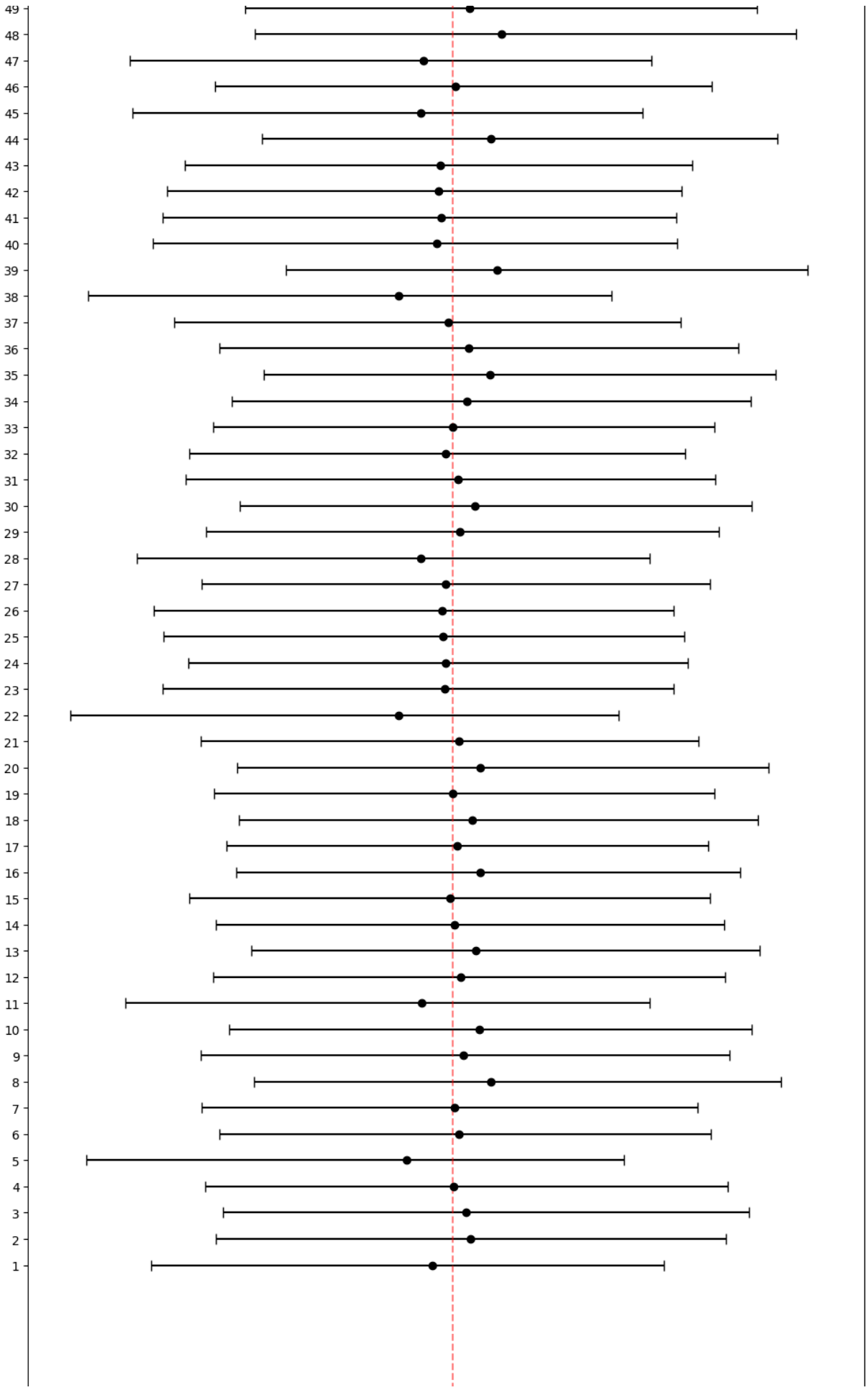
- Using a hierarchical Bayesian logistic regression we can calculate an effect for every provider against the global mean, separate from the effect of the covariates.
- This approach is relatively scalable, assesses all providers in one model, and quantifies the effect of provider relative to other covariates.

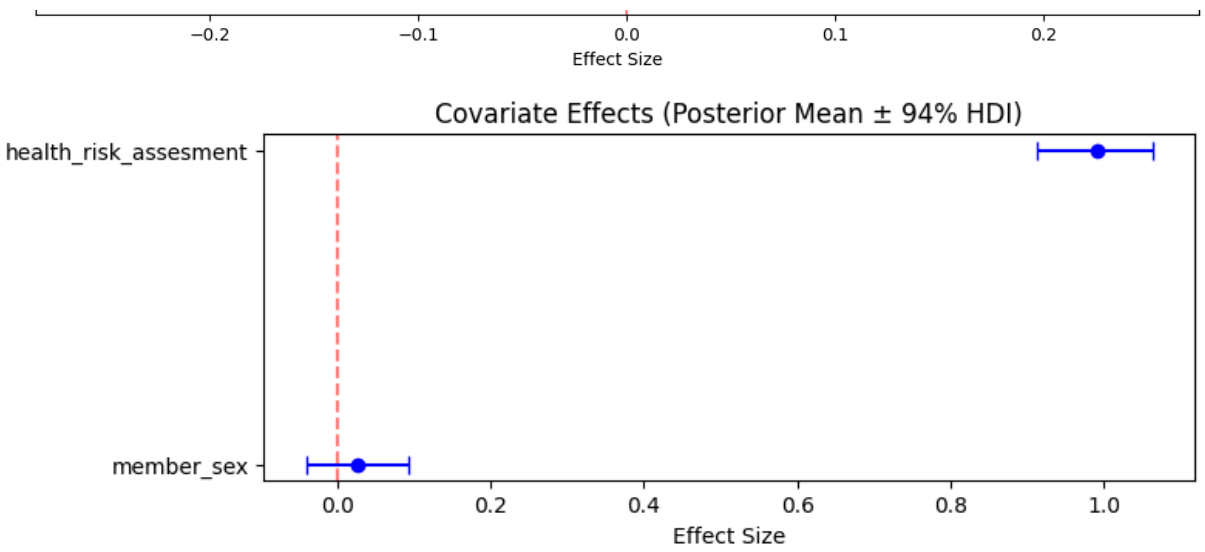
In [12]:
```
fit_data, provider_encoder = fit_hierarchical_logistic_model(
    df=data_outcome_compare,
    covariates=["member_sex","health_risk_assesment"],
    target_col="outcome",
    provider_col="servicing_provider_id"
)
```

```
[1 1 0 ... 1 1 1]
Initializing NUTS using jitter+adapt_diag...
INFO    Task(Task-3) pymc.sampling.mcmc:mcmc.py:init_nuts()- Initializing NU
TS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
INFO    Task(Task-3) pymc.sampling.mcmc:mcmc.py:sample()- Multiprocess sampl
ing (4 chains in 4 jobs)
NUTS: [sigma_provider, provider_offset, intercept, betas]
INFO    Task(Task-3) pymc.sampling.mcmc:mcmc.py:_print_step_hierarchy()- NUT
S: [sigma_provider, provider_offset, intercept, betas]
Output()
Sampling 4 chains for 1_000 tune and 2_000 draw iterations (4_000 + 8_000 dr
aws total) took 11 seconds.
INFO    Task(Task-3) pymc.sampling.mcmc:mcmc.py:_sample_return()- Sampling 4
chains for 1_000 tune and 2_000 draw iterations (4_000 + 8_000 draws total)
took 11 seconds.
```

In [13]:
```
plot_bayesian_coefficients(fit_data, provider_encoder, ["member_sex", "healt
```


Provider Effects (Posterior Mean ± 94% HDI)
96
95
94
93
92
91
90
89
88
87
86
85
84
83
82
81
80
79
78
77
76
75
74
73
72
71
70
69
68
67
66
65
64
63
62
61
60
59
58
57
56
55
54
53
52
51
50


49
48
47
46
45
44
43
42
41
40
39
38
37
36
35
34
33
32
31
30
29
28
27
26
25
24
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1



Results and interpretation of Hierarchical Bayes:

- Looking at the results from our hierarchical logistic regression, we can see that all the HDI (highest density interval) for all providers covers 0 and no provider has an average effect greater than 0.1.
- While certain providers do have an average effect greater or above 0, the width of the intervals suggests there is no real effect of any given provider on the probability of the outcome (pass or failure).
- By comparison, the health risk assessment effect has a beta of ~1, indicating a large effect on outcome.
- Member sex has a smaller effect, similar in magnitude to the provider effects.

Method 2 : Covariate Adjusted Empirical Null

- For each provider, train a logistic regression model on other providers' patients predicting outcome by covariate alone. Use this model to determine a probability of failure for each patient of this provider based on covariates alone. Use these probabilities to inform a binomial distribution for n permutations to construct a distribution of expected failure proportions for each physician given their patient covariates. Compare the physician's actual failure rate to the constructed null to determine if their failure rate is significantly deviated from the expected.
- Essentially, this analysis seeks to answer the question: Is the number of failures for a given physician significantly different from what would be expected given their patients' covariates?
- This approach is computationally expensive as it employs permutation per provider. It also quantifies and qualifies the effect a physician has on their patients given their patients' covariates.

In [14]:
```
empirical_null_covariate_test=empirical_null_provider_failures(data_outcome_
```

```
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 115.06it/
s]
```

In [15]:
```
fig = px.scatter(empirical_null_covariate_test,
    x="observed_failure_rate",
    y="expected_mean",
    color="p_value",
    title="Observed vs Expected Failure Rate by Provider",
    labels={"observed_failure_rate": "Observed Failure Rate", "p_value": "P-
)

# Add identity line (y = x)
min_val = min(empirical_null_covariate_test["observed_failure_rate"].min(),
max_val = max(empirical_null_covariate_test["observed_failure_rate"].max(),
fig.add_shape(type="line", x0=min_val, y0=min_val, x1=max_val, y1=max_val, l
fig.update_layout(height=500, width=500)
fig.show()

fig_path = os.path.join(output_dir, "expectedxobservedfailure_notbootstrappe
pio.write_image(fig, fig_path)
```

In [16]:
```
styled_df = empirical_null_covariate_test.query("p_value < 0.05").sort_value

# Highlight rows based on observed vs expected failure rate
def highlight_row(row):
    color = 'background-color: red' if row['observed_failure_rate'] > row['e
    return [color] * len(row)
empirical_null_styled_df = styled_df.style.apply(highlight_row, axis=1)

bad_providers = empirical_null_covariate_test.query("observed_failure_rate >
print(f"Providers with observed failure rate greater than expected mean: {ba
empirical_null_styled_df
```

```
Providers with observed failure rate greater than expected mean: [47, 56]
```

Out[16]:

	provider	observed_failure_rate	expected_mean	lower_95_CI	upper_95_CI	p_v
56	57	0.723404	0.589000	0.468085	0.702128	0.02
47	48	0.704545	0.580727	0.454545	0.704545	0.044
37	38	0.491803	0.599803	0.491393	0.704918	0.04
91	92	0.463415	0.600390	0.463415	0.731707	0.03
21	22	0.431818	0.571341	0.431818	0.704545	0.02
77	78	0.431818	0.579750	0.431818	0.704545	0.02
4	5	0.418605	0.542186	0.418605	0.674419	0.04
65	66	0.377778	0.544756	0.400000	0.688889	0.00
64	65	0.372093	0.501581	0.372093	0.627907	0.044

Bootstrapping for robustness

- Because we are constructing an empirical null, there is a degree of randomness to this analysis. By bootstrapping samples, we can assess the stability of the difference in distribution to have more confidence in our identification of poor performing physicians.

In [17]:
```
boostrapped_null = bootstrap_stability_check(data_outcome_compare, "outcome"

boostrapped_null_agg = boostrapped_null.groupby("provider").agg(
    observed_failure_rate=("observed_failure_rate", "median"),
    expected_mean=("expected_mean", "median"),
    p_value=("p_value", "median")
).sort_values("observed_failure_rate", ascending=False)
```

```
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.17it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 115.63it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.78it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 115.80it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 115.39it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.02it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 105.63it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.69it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 99.67it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 113.26it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 115.35it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 115.76it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.64it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.10it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 115.80it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.68it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.20it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 115.82it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 115.83it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.36it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.94it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.57it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.85it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.19it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.02it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 106.90it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.82it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.86it/
s]
```

```
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.60it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.70it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.46it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.08it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.00it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.47it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.38it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.45it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.72it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.49it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.91it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.07it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.28it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.67it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.54it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 109.14it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.95it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.44it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.47it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.60it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.96it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 115.38it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.35it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.70it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.26it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.46it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 115.69it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 115.99it/
s]
```

```
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.16it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.16it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.63it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.01it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.51it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 107.59it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.69it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 114.89it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.73it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 115.82it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 114.28it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 114.92it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.51it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.96it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.64it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 110.79it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 114.47it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 115.75it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 115.79it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.71it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.27it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.37it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.65it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.44it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 100.81it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.51it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.29it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.06it/
s]
```

```
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.71it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.19it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.55it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.28it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.28it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.22it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 106.91it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.53it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.34it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.14it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.38it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.45it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.16it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.91it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.58it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.52it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 115.82it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.87it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.86it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.24it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.27it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.36it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.40it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 108.92it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.03it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.08it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.85it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.62it/
s]
```

```
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.25it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 103.53it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.67it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.17it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.15it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.65it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.85it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.98it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 113.08it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 114.85it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.17it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 104.29it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 115.62it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.86it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.34it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.77it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.34it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.67it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.35it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.31it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.67it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.26it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.80it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.65it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.64it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.73it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 121.37it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 110.50it/
s]
```

```
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.62it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 115.89it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.72it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 110.24it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 114.49it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.88it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.60it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.76it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.23it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.26it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.53it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.11it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 101.54it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.33it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.86it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.73it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.15it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.04it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 119.07it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 106.95it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.29it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.64it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.96it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.84it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.74it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 119.18it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.01it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.49it/
s]
```

```
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.70it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.75it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.65it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.86it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.76it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.06it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 110.06it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.13it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.29it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.64it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.43it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.35it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.52it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.65it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.50it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.55it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.20it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.67it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.80it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.49it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.97it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 109.57it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 116.74it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.54it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.97it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 119.31it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.35it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.39it/
s]
```

```
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.17it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 118.21it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.38it/
s]
Simulating nulls by provider: 100%|██████████| 96/96 [00:00<00:00, 117.85it/
s]
```

In [18]:
```
fig = px.scatter(
    boostrapped_null_agg,
    x="observed_failure_rate",
    y="expected_mean",
    color="p_value",
    title="Observed vs Expected Failure Rate by Provider"+"\n"+"over Bootstr
    labels={"observed_failure_rate": "Observed Failure Rate", "p_value": "P-
)

# Add identity line (y = x)
min_val = min(boostrapped_null_agg["observed_failure_rate"].min(), boostrapp
max_val = max(boostrapped_null_agg["observed_failure_rate"].max(), boostrapp
fig.add_shape(type="line", x0=min_val, y0=min_val, x1=max_val, y1=max_val, l
fig.update_layout(height=500, width=500)
fig.show()

fig_path = os.path.join(output_dir, "expectedxobservedfailure_bootstrapped.p
pio.write_image(fig, fig_path)
```

In [19]:
```
styled_df = boostrapped_null_agg.query("p_value < 0.05").sort_values("observ

# Highlight rows based on observed vs expected failure rate
def highlight_row(row):
    color = 'background-color: red' if row['observed_failure_rate'] > row['e
    return [color] * len(row)
boostrapped_null_agg_styled_df = styled_df.style.apply(highlight_row, axis=1

bad_providers_bootsrap = boostrapped_null_agg.query("observed_failure_rate >
print(f"Providers with observed failure rate greater than expected mean: {ba
boostrapped_null_agg_styled_df
```

```
Providers with observed failure rate greater than expected mean: [57]
```

Out[19]:

provider	observed_failure_rate	expected_mean	p_value
57	0.723404	0.589722	0.022977
38	0.491228	0.599338	0.045954
92	0.453463	0.602869	0.026973
78	0.436699	0.578083	0.026474
22	0.431373	0.571168	0.028472
5	0.418393	0.538986	0.046953
89	0.415230	0.543421	0.045954
65	0.373941	0.502332	0.041958
66	0.371460	0.546906	0.006993

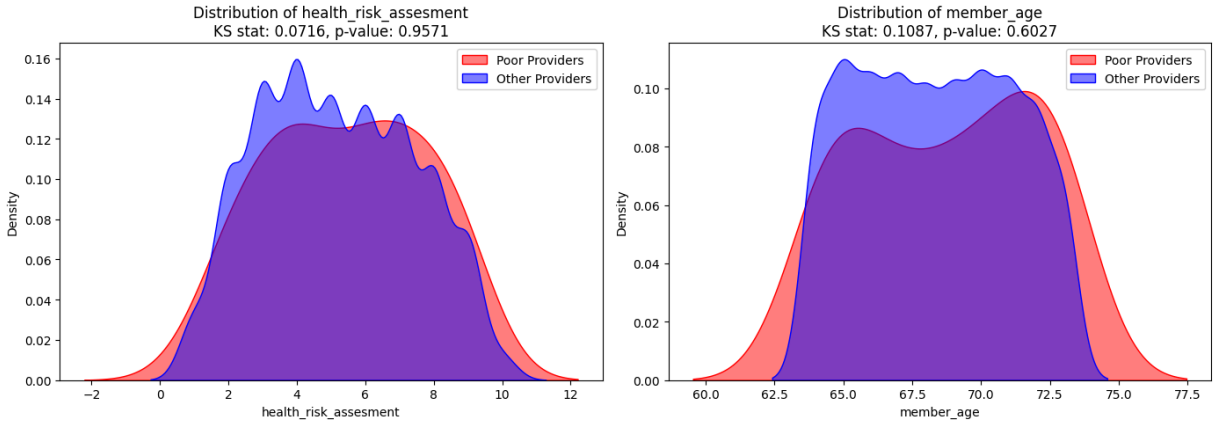
Bootstrapping identifies physicians whose observed failure rates are consistently different from expected

Before taking any actions with regard to these findings, let's confirm that we have no other information that might explain these failure rates

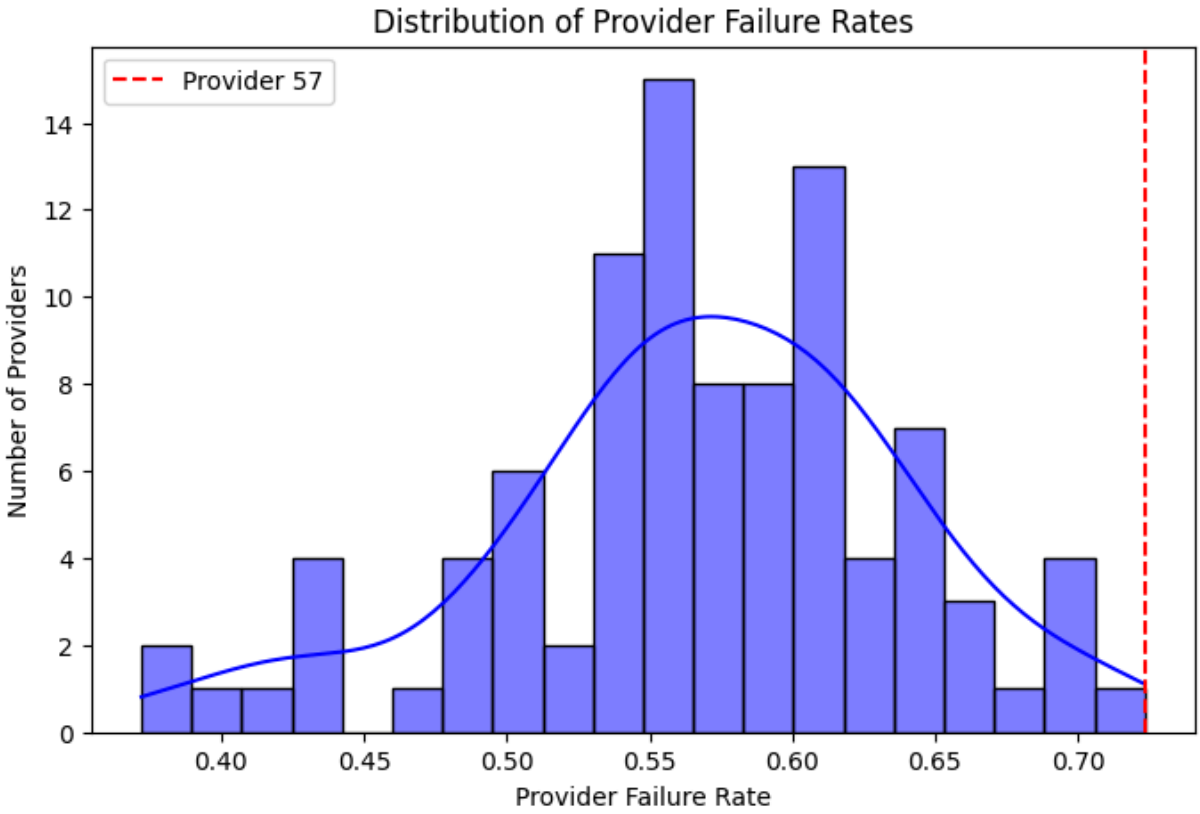
- No difference in distribution of covariates between high and low performing physicians
- Poor performing physicians identified by empirical null method have some of the highest failure rates among physicians

In [20]:
```
compare_covariate_distributions_by_provider_quality(data_outcome_compare, ba
```

```
KS statistic for health_risk_assesment: 0.0716, p-value: 0.9571
KS statistic for member_age: 0.1087, p-value: 0.6027
```



In [21]:
```
failure_rate_dist = plot_failure_rates(data_outcome_compare, bad_providers_b
```



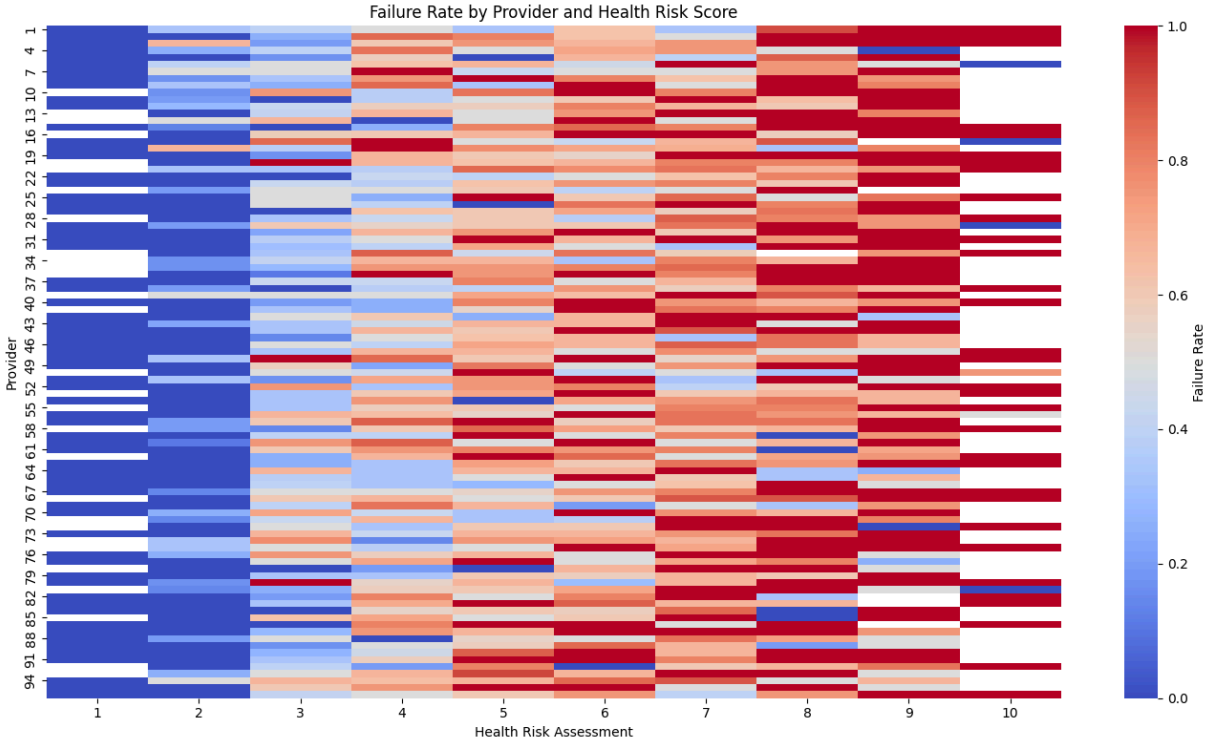
Results and interpretation of Empirical Null Analysis:

- Most providers who deviate significantly from the expected failure rate actually have reduced failure rates, suggesting these providers are performing better than expected given their patients' gender and health risk assessment.
- This makes sense in the context of our previous model in which we determined no provider had a large effect on probability of failing outcome, especially when compared to health risk assessment.
- One provider has significantly increased failure rates given their patients' covariates when examined via bootstrapped empirical null analysis.

Recommendation on reaching out to providers:

- Given the results of both the hierarchical model and the empirical null comparison, I would be cautious about recommending physician-specific intervention at this moment. While the bootstrapped empirical null did identify some providers with a significantly increased failure rate, these providers account for a very small proportion of the population of physicians, and there are other external factors that may explain these increased failure rates that we do not have insight into at this time.
- Visually below we can see that the failure rate increases relatively consistently with health risk assessment compared to provider.

In [22]:
```
plot_outcome_by_physician_hra(data_outcome_compare)
```



What data could we use to improve our recommendations? How would that change our analysis?

2. What other data would be helpful in understanding which doctors to reach out to? What other data would help you evaluate the overall clinical effectiveness of the doctor? How would you use this data?

Other data we would want access to

- Zip code
  - SDOH have been found to be very impactful on health outcomes. Zip code can be mapped to an ADI, which would give insight into the socioeconomic status of different patients.
- Treatment location
  - Where are individuals receiving these treatments? Where are these physicians practicing? If care begins in an emergent situation rather than through a proactive or preventative visit, that may also give some insight into the acuity of the conditions beyond their general health assessment.
- Time/Date of Event or Last Follow-up
  - In the analyses above, we made the assumption that individuals with no failure were healthy despite these values being NA in the original data cut. There is a

chance these individuals just haven't experienced the event yet. If we knew the dates of the events and the time since the last follow-up for individuals we don't have event data for, we could investigate whether or not provider has an effect on the time to event, especially of interest if this is a chronic condition.

- Disease duration
  - If certain physicians focus on certain disease stages, there might be a bias in the data (i.e., if patients are more progressed in their disease, they may be more likely to have a failure, so if a physician exclusively sees more progressed patients, their failure rates may be inflated).
- Medications/Therapeutics
  - We have no insight into how the physicians are managing this disease condition; it might be that patients are electing for different treatments which is affecting their outcomes.
- Other health factors: BMI, smoking, alcohol consumption (unclear if these are incorporated into the health risk assessment)

Ex. of advatageous data : Synthesizing Follow up/Event times

- Synthesizing time to event and followup timestamps to explore how this information might change our analysis if we had access to it

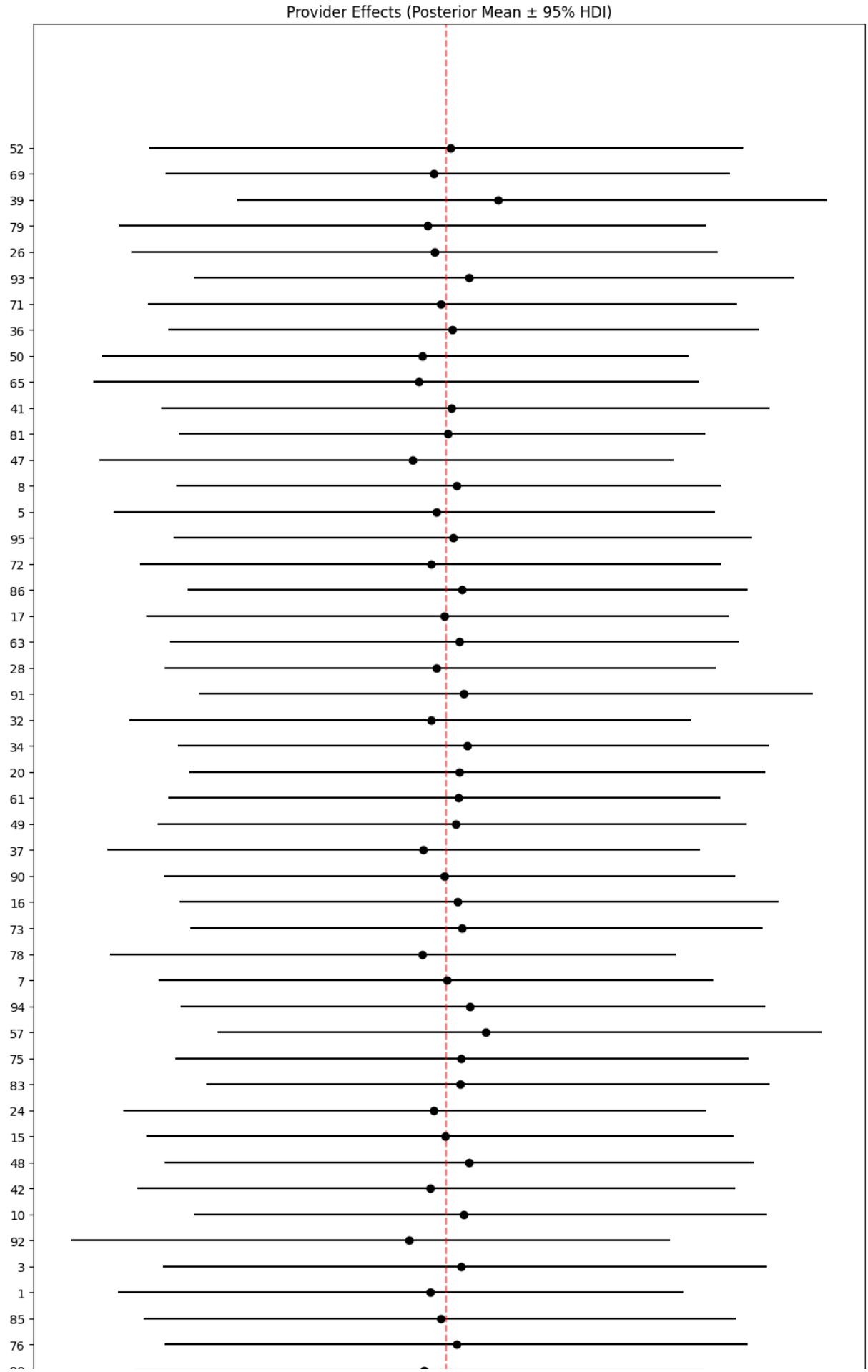
In [23]:
```
data_time_synth = generate_later_timestamp_column(data, "treatment_date", ne
```

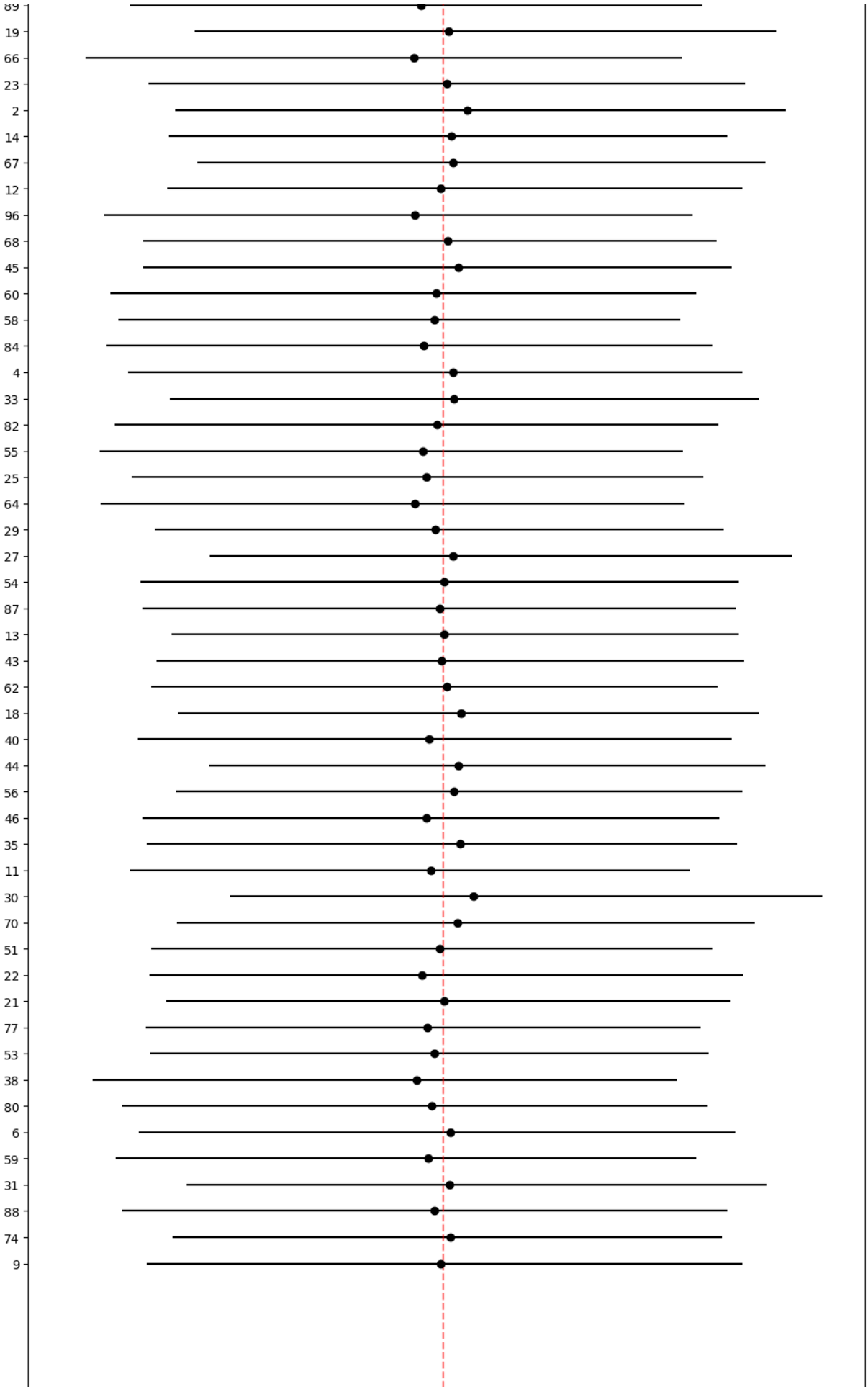
In [24]:
```
coxmodel, coxdata, provider_names = bayesian_cox_model(data_time_synth, "tre
```

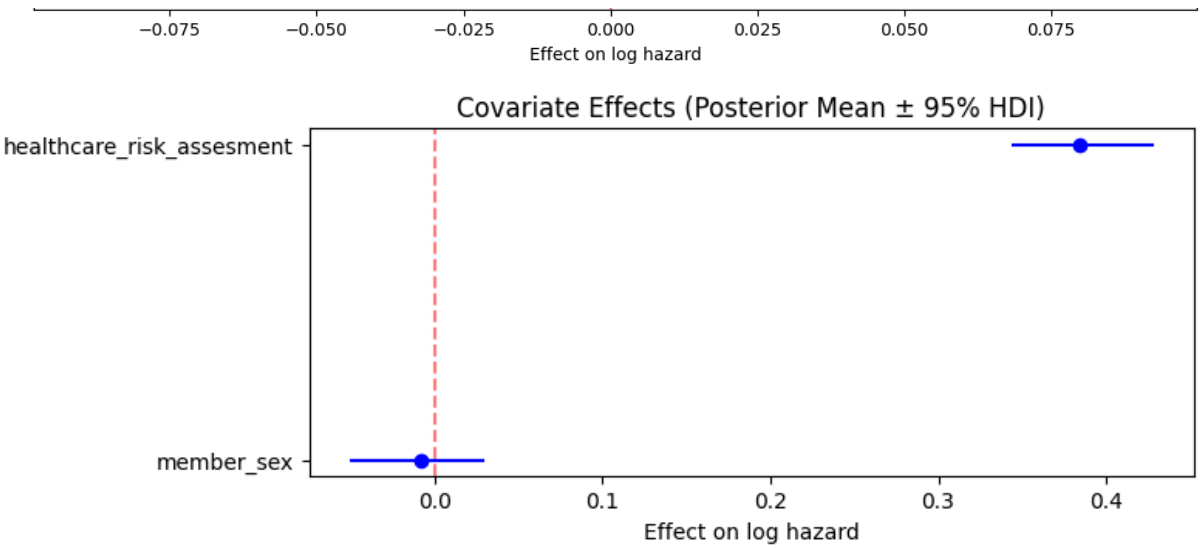
```
Initializing NUTS using jitter+adapt_diag...
INFO    Task(Task-3) pymc.sampling.mcmc:mcmc.py:init_nuts()- Initializing NU
TS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
INFO    Task(Task-3) pymc.sampling.mcmc:mcmc.py:sample()- Multiprocess sampl
ing (4 chains in 4 jobs)
NUTS: [log_h0, beta, sigma_provider, provider_offset]
INFO    Task(Task-3) pymc.sampling.mcmc:mcmc.py:_print_step_hierarchy()- NUT
S: [log_h0, beta, sigma_provider, provider_offset]
Output()
```

```
Sampling 4 chains for 1_000 tune and 1_000 draw iterations (4_000 + 4_000 dr
aws total) took 4 seconds.
INFO    Task(Task-3) pymc.sampling.mcmc:mcmc.py:_sample_return()- Sampling 4
chains for 1_000 tune and 1_000 draw iterations (4_000 + 4_000 draws total)
took 4 seconds.
```

In [25]:
```
plot_bayesian_cox_effects(coxdata, provider_names, ["member_sex", "healthcar
```


Provider Effects (Posterior Mean ± 95% HDI)
52
69
39
79
26
93
71
36
50
65
41
81
47
8
5
95
72
86
17
63
28
91
32
34
20
61
49
37
90
16
73
78
7
94
57
75
83
24
15
48
42
10
92
3
1
85
76


89
19
66
23
2
14
67
12
96
68
45
60
58
84
4
33
82
55
25
64
29
27
54
87
13
43
62
18
40
44
56
46
35
11
30
70
51
22
21
77
53
38
80
6
59
31
88
74
9



Results and Interpretation :

- Simulating follow up time and event times we can see how we could assess the provider effect with regard to time to event which would allow us to answer the question do providers have an effect of the time to failure or event.
- Based on simulation data we would conclude provider does not have a significant effect however this is syntehsized data and having real follow up information may well change this conclusion.