

```
pip install lifelines
```

Start coding or [generate](#) with AI.

Setup and Data Preparation (CPIS ≥ 6 Survival Definition)

This cell sets up the environment, loads the data, and defines the core survival variables for the Cox model (time to CPIS ≥ 6).

```
# Cell 1: Setup and Data Preparation (CPIS >= 6 Survival Definition)
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import re
import os

from lifelines import KaplanMeierFitter, CoxPHFitter
from lifelines.statistics import logrank_test
from lifelines.plotting import add_at_risk_counts

# --- Data Loading ---
try:
    data_path = os.path.join("../", "data", "raw_data.xlsx")
    df = pd.read_excel('/content/Data form Chlorhexidine Trial.xlsx')
    print(f"Data loaded successfully from {data_path}")
except FileNotFoundError:
    raise FileNotFoundError("raw_data.xlsx not found. Ensure it is in the '../data/' directory.")

# --- 1. Identify Key Columns and Survival Definitions ---
cpis_cols = [c for c in df.columns if re.search(r"^\d+ CPIS Day", c)]
age_col = [c for c in df.columns if c.lower().startswith("age")][1]
apache_col = [c for c in df.columns if c.lower().startswith("apache")][1]
gender_col = [c for c in df.columns if c.lower().startswith("gender")][1]
trial_col = [c for c in df.columns if c.lower().startswith("trial arm")][1]

# Survival Event Definition 1 (For Cox Model): first day CPIS >= 6
def first_event_day(row, day_cols, threshold=6):
    """Returns the day index of the first CPIS score >= threshold, or None."""
    for c in sorted(day_cols, key=lambda x: int(x.split()[-1])):
        val = row[c]
        if pd.notnull(val) and val >= threshold:
            return int(c.split()[-1])
    return None

def last_observed_day(row, day_cols):
    """Returns the day index of the last non-null CPIS score."""
    last = 0
    for c in sorted(day_cols, key=lambda x: int(x.split()[-1])):
        if pd.notnull(row[c]):
            last = int(c.split()[-1])
    return last if last > 0 else None

# Calculate duration and event_observed for CPIS >= 6
duration = []
event_observed = []
for i, row in df.iterrows():
    ev_day = first_event_day(row, cpis_cols, threshold=6)
    last_day = last_observed_day(row, cpis_cols)
    if ev_day is not None:
        duration.append(ev_day)
        event_observed.append(1)
    else:
        duration.append(last_day)
        event_observed.append(0)

df["duration_to_vap_cpis"] = duration
df["vap_event_cpis"] = event_observed

# Filter out rows without usable follow-up
surv_df = df.dropna(subset=["duration_to_vap_cpis"]).copy()
```

```

# Prepare Covariates for Cox Model
if trial_col:
    surv_df["trial_arm_bin"] = (surv_df[trial_col[0]].astype(str).str.contains("2", case=False, na=False)).astype(bool)
if gender_col:
    surv_df["gender_bin"] = surv_df[gender_col[0]].map({"Male":1, "Female":0})

covariate_cols = ["trial_arm_bin", "gender_bin"]
if age_col: covariate_cols.append(age_col[0])
if apache_col: covariate_cols.append(apache_col[0])

# --- Output statistics for sanity check ---
T_cpis = surv_df["duration_to_vap_cpis"]
E_cpis = surv_df["vap_event_cpis"]
print("\n--- Cohort Summary (CPIS >= 6 Definition) ---")
print(f"Total N for KM analysis: {len(surv_df)}")
print(f"Total Events (CPIS >= 6): {E_cpis.sum()}")

```

Overall Kaplan-Meier Plot (Using Final Outcome Label)

This cell performs the overall survival analysis using the less granular Final Outcome column, as specified in your request.

```

# Cell 2: Overall Kaplan-Meier Plot (Using Final Outcome Label)

outcome_col = [c for c in df.columns if c.lower().startswith("outcome of the current episode")][:-1]
if not outcome_col:
    raise ValueError("The 'Outcome of the current episode' column is missing and required for this analysis.")
outcome_col_name = outcome_col[0]

# --- Define Survival Variables (Time and Event) ---
# Time (T): Last day a CPIS score was recorded (days of follow-up)
def last_observed_cpis_day(row, day_cols):
    last = 0
    for c in sorted(day_cols, key=lambda x: int(x.split()[-1])):
        if pd.notnull(row[c]):
            last = int(c.split()[-1])
    return last if last > 0 else np.nan

df['duration_for_outcome'] = df.apply(lambda row: last_observed_cpis_day(row, cpis_cols), axis=1)

# Event (E): 1 if final outcome is VAP, 0 otherwise (Censored)
df['vap_event_final'] = df[outcome_col_name].astype(str).str.upper().str.strip().apply(
    lambda x: 1 if 'VAP' in x else 0
)

# --- Filter Data for Analysis ---
surv_df_outcome = df.dropna(subset=['duration_for_outcome']).copy()

T_outcome = surv_df_outcome["duration_for_outcome"]
E_outcome = surv_df_outcome["vap_event_final"]

print(f"Total N for KM analysis (Overall Cohort - Final Outcome): {len(surv_df_outcome)}")
print(f"Number of VAP events (Final Outcome): {E_outcome.sum()}")

# --- Overall Kaplan-Meier Estimation and Plot ---
kmf_overall_outcome = KaplanMeierFitter(label="Overall VAP-free Survival (Final Outcome)")
kmf_overall_outcome.fit(T_outcome, E_outcome)

fig, ax = plt.subplots(figsize=(10, 7))

# Plot the single survival function
kmf_overall_outcome.plot_survival_function(
    ax=ax,
    ci_show=True, # Show confidence interval
    censor_styles={'marker': '+', 'ms': 10}, # Customize censoring marks
    linewidth=3
)

# Add median survival time to the plot
median_time = kmf_overall_outcome.median_survival_time_
plt.axhline(0.5, color='r', linestyle='--', label=f'Median Survival: {median_time:.0f} Days')

add_at_risk_counts(kmf_overall_outcome, ax=ax)

```

```

plt.title("Overall Kaplan-Meier: Time to VAP (Using Final Outcome Label)")
plt.xlabel("Days of Follow-up")
plt.ylabel("VAP-free Survival Probability S(t)")
plt.legend(loc='lower left')
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()

# Print key statistics
print("\n--- Overall Survival Statistics (Final Outcome) ---")
print(f"Median VAP-free time (days): {median_time:.0f}")
try:
    prob_at_10 = kmf_overall_outcome.survival_function_.loc[10].iloc[0].round(4)
    print(f"Survival probability at Day 10: {prob_at_10}")
except KeyError:
    print("Survival probability at Day 10: N/A (Follow-up did not reach 10 days for all data)")

```

▼ Life Table Summary

This code calculates and prints a life table summary based on the overall cohort's VAP-free survival using the CPIS ≥ 6 event definition.

```

# --- LIFETABLE GENERATION (using the CPIS >= 6 definition) ---

# Calculate overall KM fit again for lifetable generation convenience
# T_cpis and E_cpis are defined in Cell 1
kmf_overall_cpis = KaplanMeierFitter(label="Overall Cohort (CPIS >= 6)")
kmf_overall_cpis.fit(T, E) # T and E are set to T_cpis and E_cpis in Cell 3

print("\n--- LIFETABLE SUMMARY (Overall VAP-free Survival - CPIS >= 6 Definition) ---")
# The survival function already contains the essential components of a life table:
# 'at_risk', 'observed', 'censored', 'survival_function'.
lifetable_summary = pd.DataFrame({
    'At Risk (Start of Interval)': kmf_overall_cpis.event_table['at_risk'],
    'Events (VAP)': kmf_overall_cpis.event_table['observed'],
    'Censored': kmf_overall_cpis.event_table['censored'],
    'Survival Probability S(t)': kmf_overall_cpis.survival_function_.iloc[:, 0].round(4)
})
# Drop rows where At Risk is zero (end of study period)
lifetable_summary = lifetable_summary[lifetable_summary['At Risk (Start of Interval)'] > 0]
print(lifetable_summary.head(15).to_markdown(floatfmt=".4f"))
print(f"\n...Table truncated after 15 days for display purposes.")

```

▼ Kaplan-Meier Comparison Plots (CPIS ≥ 6 Definition)

This cell uses the more rigorous CPIS ≥ 6 definition to compare the two treatment arms.

```

# Cell 3: Kaplan-Meier Comparison Plots (CPIS >= 6 Definition)

# Data is already prepared in Cell 1 (T_cpis, E_cpis, surv_df)
T = surv_df["duration_to_vap_cpis"]
E = surv_df["vap_event_cpis"]
ix_g1 = (surv_df["trial_arm_bin"] == 0) # 0.12%
ix_g2 = (surv_df["trial_arm_bin"] == 1) # 0.2%

kmf_g1 = KaplanMeierFitter(label="Group 1 (0.12% Chlorhexidine)")
kmf_g2 = KaplanMeierFitter(label="Group 2 (0.2% Chlorhexidine)")
kmf_g1.fit(T.loc[ix_g1], E.loc[ix_g1])
kmf_g2.fit(T.loc[ix_g2], E.loc[ix_g2])

print(f"\nGroup 1 (0.12%): N={ix_g1.sum()}, Events={E.loc[ix_g1].sum()}")
print(f"Group 2 (0.2%): N={ix_g2.sum()}, Events={E.loc[ix_g2].sum()}")


# Plot 3a: Survival Comparison Plot (S(t))
fig, ax = plt.subplots(figsize=(10, 6))
kmf_g1.plot_survival_function(ax=ax, ci_show=False)
kmf_g2.plot_survival_function(ax=ax, ci_show=False)
add_at_risk_counts(kmf_g1, kmf_g2, ax=ax)
plt.title("Kaplan-Meier: VAP-free Survival by Chlorhexidine Concentration (CPIS \u2265 6)")

```

```

plt.xlabel("Days to CPIS \u2265 6")
plt.ylabel("VAP-free Survival Probability S(t)")
plt.legend(loc='lower left')
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()

# Plot 3b: Cumulative Probability of Event Plot (1 - S(t))
fig, ax = plt.subplots(figsize=(10, 6))
kmf_g1.plot_cumulative_density(ax=ax, ci_show=False)
kmf_g2.plot_cumulative_density(ax=ax, ci_show=False)
plt.title("Cumulative Probability of VAP (1 - S(t)) by Concentration (CPIS \u2265 6)")
plt.xlabel("Days to CPIS \u2265 6")
plt.ylabel("Cumulative Probability of VAP")
plt.legend(loc='upper left')
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()

```

▼ Cox Proportional Hazards Model and Hazard Ratios

This cell fits the multivariate CPH model.

```

# Cell 4: Cox Proportional Hazards Model and Hazard Ratios

# --- Prepare Data for CoxPH ---
cox_df = surv_df[["duration_to_vap_cpis", "vap_event_cpis"] + covariate_cols].copy()
cox_df = cox_df.dropna()
cox_df = cox_df.rename(columns={"duration_to_vap_cpis": "duration", "vap_event_cpis": "event"})

# --- Cox Proportional Hazards (CPH) Model Fitting ---
cph = CoxPHFitter()
cph.fit(cox_df, duration_col="duration", event_col="event", show_progress=True)

print("\n--- Cox Proportional Hazards Model Summary (Hazard Ratios) ---")
cph.print_summary(decimals=4)

# Forest plot of hazard ratios
fig, ax = plt.subplots(figsize=(8,5))
cph.plot(hazard_ratios=True, ax=ax)
plt.title("Covariate Hazard Ratios (HR) and 95% CI for VAP (CPIS \u2265 6)")
plt.grid(True, axis='x', alpha=0.3)
plt.tight_layout()
plt.show()

```

▼ Adjusted Survival Curves and PH Assumption Check

This final cell validates the model and shows covariate effects.

```

# Cell 5: Adjusted Survival Curves and PH Assumption Check

# --- Adjusted Survival Curves (Partial Effects Plots) ---
print("\n--- Adjusted Survival Curves: Partial Effects Plots ---")

# a) Adjusted VAP-free survival by Trial Arm
if "trial_arm_bin" in cox_df.columns:
    fig, ax = plt.subplots(figsize=(10,6))
    cph.plot_partial_effects_on_outcome(
        covariates="trial_arm_bin",
        values=[0, 1],
        cmap='coolwarm',
        ax=ax
    )
    plt.title("Adjusted VAP-free Survival by Trial Arm (0.12% vs 0.2%)")
    plt.xlabel("Days")
    plt.ylabel("Adjusted Survival Probability")
    plt.grid(True)
    plt.show()

# b) Adjusted VAP-free survival by Age (if available)
if age_col and age_col[0] in cox_df.columns:

```

```
fig, ax = plt.subplots(figsize=(10,6))
cph.plot_partial_effects_on_outcome(
    covariates=age_col[0],
    values=np.arange(40, 80, 10), # Plot survival for ages 40, 50, 60, 70
    cmap='viridis',
    ax=ax
)
plt.title("Adjusted VAP-free Survival by Age")
plt.xlabel("Days")
plt.ylabel("Adjusted Survival Probability")
plt.grid(True)
plt.show()

# --- Proportional Hazards (PH) Assumption Check ---
print("\n--- Proportional Hazards Assumption Check (Schoenfeld Residuals) ---")
# This command automatically generates plots for the check.
cph.check_assumptions(cox_df, show_plots=True)
```

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.