# Suppplemental Material 1: Code

## R-Code

Python:

#mixed effect logistische regression mit lme4

df <- read.csv("XXXX")

library(janitor)

df <- janitor::clean\_names(df)

df$zeit <- as.difftime(df$zeit)

library(lme4)

df$ecg\_change <- as.factor(df$ecg\_change)

library(janitor)

df <- clean\_names(df)

str(df)

model <- glmer(

 ecg\_change ~ sdnn\_ms + r\_mssd\_ms + p\_nn50 + (1 | proband),

 data = df,

 family = binomial

)

summary(model)

# Calculate the Odds Ratios (OR)

odds\_ratios <- exp(fixef(model))

# Calculate the 95% confidence intervals for the fixed effects using the "Wald" method

conf\_ints <- confint(model, method = "Wald", level = 0.95)

# Exponentiate the confidence intervals to get the CI for odds ratios

conf\_ints\_OR <- exp(conf\_ints)

# Create a data frame of Odds Ratios and Confidence Intervals

print(names(odds\_ratios))

print(odds\_ratios)

print(conf\_ints\_OR)

# Forest Plot

library(broom.mixed)

library(ggplot2)

library(dplyr)

# Tidy the model output

tidy\_model <- tidy(model, effects = "fixed", conf.int = TRUE, exponentiate = TRUE)

# Rename terms for clarity

tidy\_model <- tidy\_model %>%

 mutate(term = recode(term,

 "sdnn\_ms" = "SDNN",

 "r\_mssd\_ms" = "RMSSD",

 "p\_nn50" = "pNN50"

 ))

# Filter and order terms for plotting

tidy\_model <- tidy\_model %>%

 filter(term != "(Intercept)") %>%

 arrange(desc(estimate)) %>%

 mutate(term = factor(term, levels = term)) # Keep order in plot

# Highlight significant effects

tidy\_model <- tidy\_model %>%

 mutate(Significance = ifelse(conf.low > 1 | conf.high < 1, "Significant", "Not significant"))

# Forest plot

ggplot(tidy\_model, aes(x = estimate, y = term, color = Significance)) +

 geom\_point(size = 3) +

 geom\_errorbarh(aes(xmin = conf.low, xmax = conf.high), height = 0.2, linewidth = 1) +

 geom\_vline(xintercept = 1, linetype = "dashed", color = "grey40") +

 scale\_x\_log10() + # Optional: use log scale for better visualization of odds ratios

 scale\_color\_manual(values = c("Significant" = "#D55E00", "Not significant" = "grey40")) +

 labs(

 title = "Association between HRV Parameters and ECG Changes",

 x = "Odds Ratio (log scale, 95% CI)",

 y = "HRV Metric"

 ) +

 theme\_minimal(base\_size = 14) +

 theme(

 axis.text.y = element\_text(size = 13),

 axis.title = element\_text(size = 14),

 plot.title = element\_text(size = 16, face = "bold", hjust = 0.5),

 legend.title = element\_blank(),

 legend.position = "bottom"

 )

# on mission level

df2 <- read.csv("XXX")

str(df2)

library(janitor)

df2 <- clean\_names(df2)

str(df2)

library(lme4)

model2 <- glmer(

 bin\_ecg ~ sdnn\_overall\_mean + pnn50\_overall\_mean + rmssd\_overall\_mean + (1 | prob\_id),

 data = df2,

 family = binomial

)

summary(model2)

# Calculate the Odds Ratios (OR)

odds\_ratios <- exp(fixef(model2))

# Calculate the 95% confidence intervals for the fixed effects using the "Wald" method

conf\_ints <- confint(model2, method = "Wald", level = 0.95)

# Exponentiate the confidence intervals to get the CI for odds ratios

conf\_ints\_OR <- exp(conf\_ints)

# Ensure that the odds\_ratios and the confidence intervals align

# Create a data frame of Odds Ratios and Confidence Intervals

print(names(odds\_ratios))

print(odds\_ratios)

print(conf\_ints\_OR)

## Forenst plot

library(broom.mixed)

library(ggplot2)

library(dplyr)

# Tidy the model output (fixed effects only)

tidy\_model2 <- tidy(model2, effects = "fixed", conf.int = TRUE, exponentiate = TRUE)

# Rename HRV variables for readability

tidy\_model2 <- tidy\_model2 %>%

 mutate(term = recode(term,

 "sdnn\_overall\_mean" = "SDNN",

 "rmssd\_overall\_mean" = "RMSSD",

 "pnn50\_overall\_mean" = "pNN50"

 ))

# Filter, order, and format

tidy\_model2 <- tidy\_model2 %>%

 filter(term != "(Intercept)") %>%

 arrange(desc(estimate)) %>%

 mutate(term = factor(term, levels = term),

 Significance = ifelse(conf.low > 1 | conf.high < 1, "Significant", "Not significant"))

# Create forest plot

ggplot(tidy\_model2, aes(x = estimate, y = term, color = Significance)) +

 geom\_point(size = 3) +

 geom\_errorbarh(aes(xmin = conf.low, xmax = conf.high), height = 0.2, linewidth = 1) +

 geom\_vline(xintercept = 1, linetype = "dashed", color = "grey40") +

 scale\_x\_log10() + # Log scale for odds ratio

 scale\_color\_manual(values = c("Significant" = "#D55E00", "Not significant" = "grey40")) +

 labs(

 title = "Association Between HRV Metrics and ECG Changes per Mission",

 x = "Odds Ratio (log scale, 95% CI)",

 y = "HRV Metric"

 ) +

 theme\_minimal(base\_size = 14) +

 theme(

 axis.text.y = element\_text(size = 13),

 axis.title = element\_text(size = 14),

 plot.title = element\_text(size = 16, face = "bold", hjust = 0.5),

 legend.title = element\_blank(),

 legend.position = "bottom"

 )

## supplement with sex:

df <- read.csv("XXX")

library(janitor)

df <- janitor::clean\_names(df)

df$zeit <- as.difftime(df$zeit)

library(lme4)

df$ecg\_change <- as.factor(df$ecg\_change)

df <- clean\_names(df)

str(df)

model <- glmer(

 ecg\_change ~ sdnn\_ms + r\_mssd\_ms + p\_nn50 + sex + (1 | proband),

 data = df,

 family = binomial

)

summary(model)

# Calculate the Odds Ratios (OR)

odds\_ratios <- exp(fixef(model))

# Calculate the 95% confidence intervals for the fixed effects using the "Wald" method

conf\_ints <- confint(model, method = "Wald", level = 0.95)

# Exponentiate the confidence intervals to get the CI for odds ratios

conf\_ints\_OR <- exp(conf\_ints)

# Create a data frame of Odds Ratios and Confidence Intervals

print(names(odds\_ratios))

print(odds\_ratios)

print(conf\_ints\_OR)

## supplement with heart rate:

df <- read.csv("XXX")

library(janitor)

df <- janitor::clean\_names(df)

df$zeit <- as.difftime(df$zeit)

library(lme4)

df$ecg\_change <- as.factor(df$ecg\_change)

df <- clean\_names(df)

str(df)

library(lme4)

model <- glmer(

 ecg\_change ~ sdnn\_ms + r\_mssd\_ms + p\_nn50 + mittel\_bpm + (1 | proband),

 data = df,

 family = binomial

)

summary(model)

# Calculate the Odds Ratios (OR)

odds\_ratios <- exp(fixef(model))

# Calculate the 95% confidence intervals for the fixed effects using the "Wald" method

conf\_ints <- confint(model, method = "Wald", level = 0.95)

# Exponentiate the confidence intervals to get the CI for odds ratios

conf\_ints\_OR <- exp(conf\_ints)

# Ensure that the odds\_ratios and the confidence intervals align

# Create a data frame of Odds Ratios and Confidence Intervals

print(names(odds\_ratios))

print(odds\_ratios)

print(conf\_ints\_OR)

## Python

from itertools import combinations

from scipy.stats import ttest\_rel

def run\_paired\_ttests(df, metrics, phases, alpha=0.05):

 phase\_label\_map = {

 'alarm': 'Alarm',

 'zBO': 'En-route',

 'pat\_care': 'Patient Care',

 'Transport': 'Transport'

 }

 results = []

 for metric in metrics:

 for phase1, phase2 in combinations(phases, 2):

 col1 = f'{metric}\_{phase1}\_mean'

 col2 = f'{metric}\_{phase2}\_mean'

 # Drop rows with missing values for either column

 valid\_data = df[[col1, col2]].dropna()

 # Paired t-test

 t\_stat, p\_value = ttest\_rel(valid\_data[col1], valid\_data[col2])

 results.append({

 'Metric': metric.upper(),

 'Phase 1': phase\_label\_map[phase1],

 'Phase 2': phase\_label\_map[phase2],

 't-statistic': t\_stat,

 'p-value': p\_value

 })

 # Adjust p-values using Bonferroni correction

 result\_df = pd.DataFrame(results)

 result\_df['p-value (Bonferroni)'] = result\_df['p-value'] \* len(result\_df)

 result\_df['Significant'] = result\_df['p-value (Bonferroni)'] < alpha

 return result\_df

# Define metrics and phases

metrics = ['rmssd', 'sdnn', 'pnn50']

phases = ['alarm', 'zBO', 'pat\_care', 'Transport']

# Run paired t-tests

ttest\_results = run\_paired\_ttests(crf\_hrv, metrics, phases)

# Display results

print(ttest\_results)

def get\_significant\_pairs(df, metrics, phases, alpha=0.05):

 from itertools import combinations

 from scipy.stats import ttest\_rel

 phase\_label\_map = {

 'alarm': 'Alarm',

 'zBO': 'En-route',

 'pat\_care': 'Patient Care',

 'Transport': 'Transport'

 }

 results = {}

 for metric in metrics:

 sig\_pairs = []

 all\_tests = []

 for phase1, phase2 in combinations(phases, 2):

 col1 = f'{metric}\_{phase1}\_mean'

 col2 = f'{metric}\_{phase2}\_mean'

 valid = df[[col1, col2]].dropna()

 t\_stat, p\_val = ttest\_rel(valid[col1], valid[col2])

 all\_tests.append((phase1, phase2, p\_val))

 # Bonferroni correction

 corrected = [(a, b, p, p \* len(all\_tests)) for a, b, p in all\_tests]

 sig\_pairs = [(a, b, p\_corr) for a, b, p, p\_corr in corrected if p\_corr < alpha]

 results[metric] = sig\_pairs

 return results

import matplotlib.pyplot as plt

import seaborn as sns

import pandas as pd

def plot\_merged\_boxplots\_with\_significance\_pub(df, metrics, phases, sig\_results, save\_path = None):

 phase\_label\_map = {

 'alarm': 'Alarm',

 'zBO': 'En-route',

 'pat\_care': 'Patient Care',

 'Transport': 'Transport'

 }

 metric\_label\_map = {

 'rmssd': 'RMSSD [ms]',

 'sdnn': 'SDNN [ms]',

 'pnn50': 'pNN50'

 }

 sns.set(style='whitegrid', context='notebook', font\_scale=1.2)

 plt.figure(figsize=(12, 8))

 for i, metric in enumerate(metrics, start=1):

 metric\_cols = [f'{metric}\_{phase}\_mean' for phase in phases]

 metric\_data = df[metric\_cols].copy()

 # Rename columns to phase labels for plotting

 metric\_data.columns = [phase\_label\_map[p] for p in phases]

 melted = metric\_data.melt(var\_name='Phase', value\_name='Value')

 ax = plt.subplot(1, len(metrics), i)

 sns.boxplot(

 x='Phase', y='Value', data=melted, palette="Set2",

 width=0.6, linewidth=1.5, ax=ax, fliersize=3

 )

 #ax.set\_ylim(bottom=0)

 # Set x-axis tick labels explicitly

 ax.set\_xticklabels(list(phase\_label\_map.values()), rotation=45, fontsize=12)

 # Titles and labels

 ax.set\_title(metric\_label\_map[metric], fontsize=16, weight='bold')

 ax.set\_xlabel('Mission Phase', fontsize=14)

 ax.set\_ylabel(metric\_label\_map[metric], fontsize=14)

 ax.tick\_params(labelsize=12)

 # Annotate significance

 sig\_pairs = sig\_results.get(metric, [])

 phase\_labels = [phase\_label\_map[p] for p in phases]

 y\_max = melted['Value'].max()

 y\_min = melted['Value'].min()

 spacing = (y\_max - y\_min) \* 0.05

 for idx, (p1, p2, p\_corr) in enumerate(sig\_pairs):

 x1 = phase\_labels.index(phase\_label\_map[p1])

 x2 = phase\_labels.index(phase\_label\_map[p2])

 y = y\_max + spacing \* (idx + 1)

 stars = '\*\*\*' if p\_corr < 0.001 else '\*\*' if p\_corr < 0.01 else '\*'

 ax.plot([x1, x1, x2, x2], [y, y + spacing \* 0.2, y + spacing \* 0.2, y], lw=1.5, color='black')

 ax.text((x1 + x2) / 2, y + spacing \* 0.25, stars,

 ha='center', va='bottom', fontsize=14, weight='bold')

 sns.despine(trim=True)

 ax.grid(axis='y', linestyle='--', linewidth=0.5)

 if save\_path:

 plt.savefig(save\_path, dpi=300, bbox\_inches='tight') # Save high-res figure with tight layout

 plt.tight\_layout()

 plt.show()

# Run paired t-tests

sig\_results = get\_significant\_pairs(crf\_hrv, metrics, phases)

# Plot boxplots with significance annotations

plot\_merged\_boxplots\_with\_significance\_pub(crf\_hrv, metrics, phases, sig\_results, save\_path='Figure 3\_1.png')