War Recurrence

Libraries

```r
# install.packages("readxl")
# install.packages("tidyverse")
# install.packages("ggplot2")
# install.packages("survminer")
# install.packages("timeROC")
# install.packages("caret")
# install.packages("timeROC")
# install.packages("caret")
# Install and load necessary packages
# install.packages("randomForestSRC")
# install.packages("tidyr")
#install.packages("Synth")

library(Synth)
```

```
##
## Synth Package: Implements Synthetic Control Methods.

## See https://web.stanford.edu/~jhain/synthpage.html for additional information.
```

```r
library(pec)
```

```
Loading required package: prodlim
```

```r
library(expss)
```

```
Loading required package: maditr
```

```
To aggregate all non-grouping columns: take_all(mtcars, mean, by = am)


Attaching package: 'maditr'

The following object is masked from 'package:base':

    sort_by
```

```r
library(tidyr)
```

```
Attaching package: 'tidyr'

The following objects are masked from 'package:expss':

    contains, nest
```

```r
library(caret)
```

```
Loading required package: ggplot2


Attaching package: 'ggplot2'

The following object is masked from 'package:expss':

    vars

Loading required package: lattice


Attaching package: 'caret'

The following object is masked from 'package:pec':

    R2
```

```r
library(stargazer)
```

```
Please cite as:

 Hlavac, Marek (2022). stargazer: Well-Formatted Regression and Summary Statistics Tables.

 R package version 5.2.3. https://CRAN.R-project.org/package=stargazer
```

```r
library(survival)
```

```
Attaching package: 'survival'

The following object is masked from 'package:caret':

    cluster
```

```r
library(survminer)
```

```
Loading required package: ggpubr

Attaching package: 'ggpubr'

The following object is masked from 'package:expss':

    compare_means

Attaching package: 'survminer'

The following object is masked from 'package:survival':

    myeloma
```

```r
library(car)
```

```
Loading required package: carData


Attaching package: 'car'

The following object is masked from 'package:expss':

    recode
```

```r
library(carData)
library(readxl)
library(car)
library(carData)
library(date)
library(readxl)
library(scales)
library(dplyr)
```

```

Attaching package: 'dplyr'

The following object is masked from 'package:car':

    recode

The following objects are masked from 'package:expss':

    compute, contains, na_if, recode, vars, where

The following objects are masked from 'package:maditr':

    between, coalesce, first, last

The following objects are masked from 'package:stats':

    filter, lag
```

```
The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```r
library(gridExtra)
```

```
Attaching package: 'gridExtra'

The following object is masked from 'package:dplyr':

    combine
```

```r
library(stargazer)
library(readxl)
library(car)
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v forcats   1.0.0     v readr     2.1.5
v lubridate 1.9.3     v stringr   1.5.1
v purrr     1.0.2     v tibble    3.2.1

-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::between()     masks maditr::between()
x dplyr::coalesce()    masks maditr::coalesce()
x readr::col_factor()  masks scales::col_factor()
x readr::cols()        masks maditr::cols()
x gridExtra::combine() masks dplyr::combine()
x dplyr::compute()     masks expss::compute()
x dplyr::contains()    masks tidyr::contains(), expss::contains()
x purrr::discard()     masks scales::discard()
x dplyr::filter()      masks stats::filter()
x dplyr::first()       masks maditr::first()
x stringr::fixed()     masks expss::fixed()
x purrr::keep()        masks expss::keep()
x dplyr::lag()         masks stats::lag()
x dplyr::last()        masks maditr::last()
x purrr::lift()        masks caret::lift()
x purrr::modify()      masks expss::modify()
```

```
x purrr::modify_if()   masks expss::modify_if()
x dplyr::na_if()       masks expss::na_if()
x tidyr::nest()        masks expss::nest()
x dplyr::recode()      masks car::recode(), expss::recode()
x stringr::regex()     masks expss::regex()
x purrr::some()        masks car::some()
x purrr::transpose()   masks maditr::transpose()
x dplyr::vars()        masks ggplot2::vars(), expss::vars()
x purrr::when()        masks expss::when()
x dplyr::where()       masks expss::where()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
```

```r
library(survival)
library(survminer)
library(carData)
library(readxl)
library(carData)
options(scipen=999)
library(date)
library(readxl)
library(scales)
library(ggplot2)
library(gridExtra)
library(randomForestSRC)
```

```
 randomForestSRC 3.2.3

 Type rfsrc.news() to see new features, changes, and bug fixes.



Attaching package: 'randomForestSRC'

The following object is masked from 'package:purrr':

    partial
```

```r
library(readxl)
library(pec)
library(timeROC)
```

```r
library(survivalROC)
library(ggplot2)
library(expss)
library(survival)
library(dplyr)
library(caret)
library(randomForestSRC)
library(readxl)
library(pec)
library(timeROC)
library(survivalROC)
library(ggplot2)
library(expss)
library(survival)
library(dplyr)
library(caret)
```

Data Load and Merge

Load Data

```r
episodes <- read_excel("episodes.xlsx")

episodes$polity_scaled<- scale(episodes$p_polity2, center = TRUE, scale = TRUE)
episodes$gov_ter<-ifelse(episodes$wt==1, 1,0)
episodes$gov_war<-ifelse(episodes$wt==2, 1,0)
episodes$ter_war<-ifelse(episodes$wt==3, 1,0)
episodes$coal<- scale(episodes$W4, center = TRUE, scale = TRUE)


episodess <- subset(episodes, intensity_level == 2)
episodest<- subset(episodes, type_of_conflict ==3 )

peaces<- episodes %>% filter(peace== 1)
```

Data merge- recycled

```
# episodes_subset <- select(episodes, conflict_id, outcome, recur_anyy, recur_side, ps, recu

#isq_subset <- select(ISQ_ACD, year, isq2015_id, recur)
```

```
#qog_vars <- c(
  # Military and conflict-related variables
 # "wdi_armimp",         # Arms imports (SIPRI trend indicator values)
  #"wvs_fight",          # Willingness to fight for country
  #"wdi_expmil",         # Military expenditure (% of GDP)
  #"wdi_expmilge",       # Military expenditure (% of general government expenditure)
  #"bicc_hw",            # Heavy Weapons Index
  #"bicc_gmi",           # Global Militarization Index
  #"atop_ally",          # Member of an Alliance
  #"atop_number",        # Number of Alliances
  #"bicc_milexp",        # Military Expenditure Index
  #"fe_etfra",           # Ethnic Fractionalization
  #"year",
  #"cname",

  # Economic and political variables
  #"al_ethnic2000",      # Ethnic fractionalization index, 2000
  #"al_language2000",    # Language fractionalization index, 2000
  #"al_religion2000",    # Religious fractionalization index, 2000
  #"fe_cultdiv",         # Cultural diversity index
  #"p_durable",          # Political durability
  #"wdi_gdpcapcur",      # GDP per capita (current US$)
  #"p_polity2"           # Revised polity score
#)
```

```
qog_vars <- c(
  # New variables added
  "ht_colonial",        # Historical colonialism
  "vdem_gender",        # Gender equality index
  "gle_pop",            # Population data
  "gle_gdp",            # GDP data
  "fe_plural",          # Pluralism index
  "vdem_exbribe",       # Executive bribes index
  "vdem_execorr",       # Executive corruption index
  "vdem_exembez",       # Executive embezzlement index
  "vdem_jucorrdc",      # Judicial corruption index
  "vdem_libdem",        # Liberal democracy index
```

```r
  "wdi_pop",              # Population data from World Development Indicators
  "cname",                # Country name
  "year"                  # Year
)
```

```r
#episodes<-merge(episodes, selected_qog_vars,by=c("cname","year"), all.x= TRUE)
#eps<-merge(episodes, NewWmeasure,by=c("country_name","year"), all.x= TRUE)
```

```r
#epid<-merge(episodes,tem_sean,by=c("conflict_id","end_year"), all=TRUE)

#ACD_Sean_1000<-merge(episodes_1000,ISQ_ACD,by=c("conflict_id","year"), all=TRUE)



#ACD_Sean_1000<-merge(episodes_1000merge,ISQ_ACD_Merge,by=c("isq2015_id","year"), all=TRUE)


#ep_qog<-merge(episodes,qog_std_ts_jan23,by=c("cname","year"))
#ep<-merge(copy,ep_qog,by=c("conflict_id","end_year"), all=TRUE)
#clean_ep<-ep[, sapply(ep, function(col) length(na.omit(col))) >150]



#dur_acd<-merge(duration,ucdp_brd_dyadic_221 ,by=c("cname", "year"))


#epidd$log_gdp<-log(epidd$wdi_gdpcapcur)

#fromqog= subset(qog_std_ts_jan23, select =c(cname, year,fe_etfra) )
#epid<-merge(episodes, fromqog,by=c("cname","year"))
#epidd<-merge(copy, epid,by=c("conflict_id","end_year"))



#episodes$yearr<- as.Date(episodes$start_date2)
#episodes$yea<-date.mdy(episodes$yearr)$yea
#episodes$yea<-episodes$yea+10
#episodes$end<- as.Date(episodes$ep_end_date)
#episodes$en<-date.mdy(episodes$end)$en
#episodes$en<-episodes$en+10

#X= subset(clean_o, select = -c(recur_a, conflict_id, end_year, cname, year, start_year, loca
```

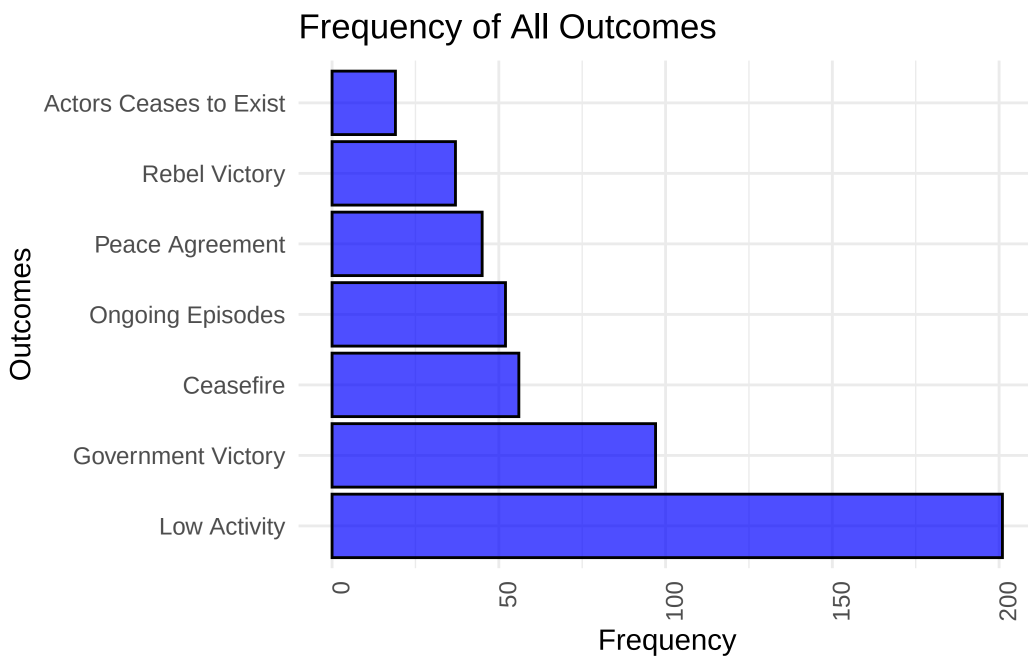
```
#Y=clean_o$recur_a
#clean_ep<-ep[, sapply(ep, function(col) length(na.omit(col))) >150]
#clean_o<-na.omit(clean_ep)

#episodes$peace<-ifelse(episodes$outcome==1, 1,0)
#episodes$cease<-ifelse(episodes$outcome==2, 1,0)
#episodes$govvic<-ifelse(episodes$outcome==3, 1,0)
#episodes$rebvic<-ifelse(episodes$outcome==4, 1,0)
#episodes$lowac<-ifelse(episodes$outcome==5, 1,0)
#episodes$dis<-ifelse(episodes$outcome==6, 1,0)
#episodes$lowcease<-Recode(episodes$outcome, "1=0; 2=1; 3=0; 4=0; 5=1; 6=0")
#episodes$log_dur<-log(episodes$duration)
```

EDA

Distribution of Outcome

```
episodes$outs <- as.character(episodes$outcome) # Convert factor to character if necessary

# Define a mapping for the outcomes
outcome_labels <- c("1" = "Peace Agreement",
                    "2" = "Ceasefire",
                    "3" = "Government Victory",
                    "4" = "Rebel Victory",
                    "5" = "Low Activity",
                    "6" = "Actors Ceases to Exist")

# Replace the coded values with the corresponding labels
episodes$outs <- sapply(episodes$outs, function(x) outcome_labels[x])

# Replace NA values with "Ongoing Episodes"
episodes$outs[is.na(episodes$outs)] <- "Ongoing Episodes"

# Convert the 'outs' column back to factor and set the desired order
episodes$outs <- factor(episodes$outs, levels = c("Peace Agreement", "Ceasefire", "Governmen

# Create the bar plot
outcome_freq <- ggplot(episodes, aes(x = reorder(outs, outs, function(x) -length(x)))) +
  geom_bar(fill = "blue", color = "black", alpha = 0.7) +
  labs(title = "Frequency of All Outcomes",
```

```r
       x = "Outcomes",
       y = "Frequency") +
  theme_minimal() +
  coord_flip() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) # Rotate x-axis labels for better

# Display the plot
print(outcome_freq)
```



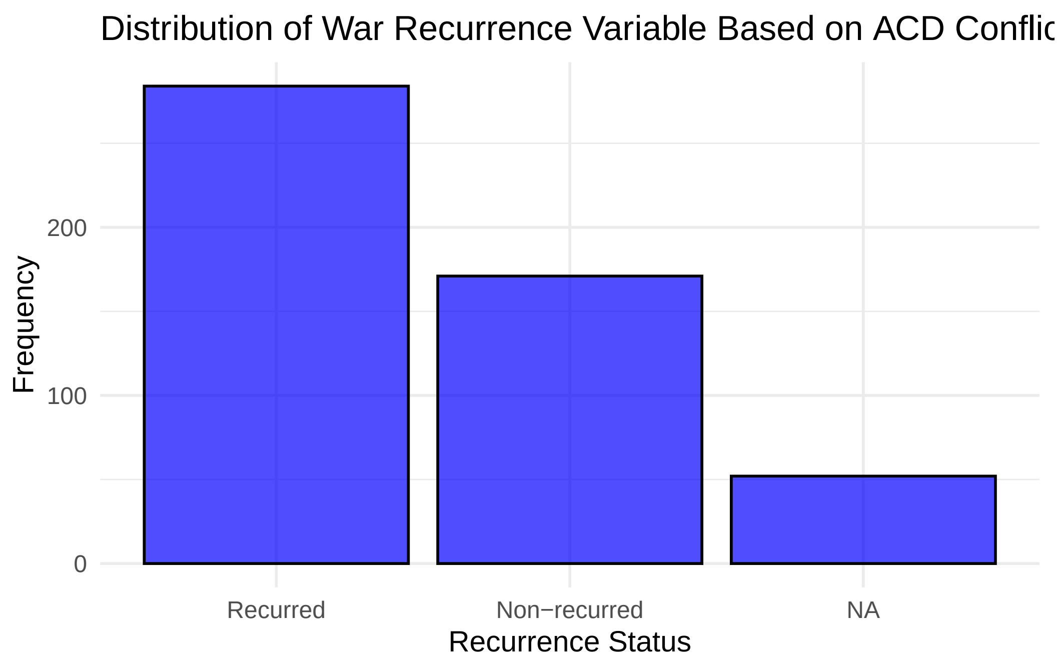
Cross-Tab Viz

```r
# Filter out rows with NA in the recurr column
episodes |>
  filter(!is.na(recur_any))
```

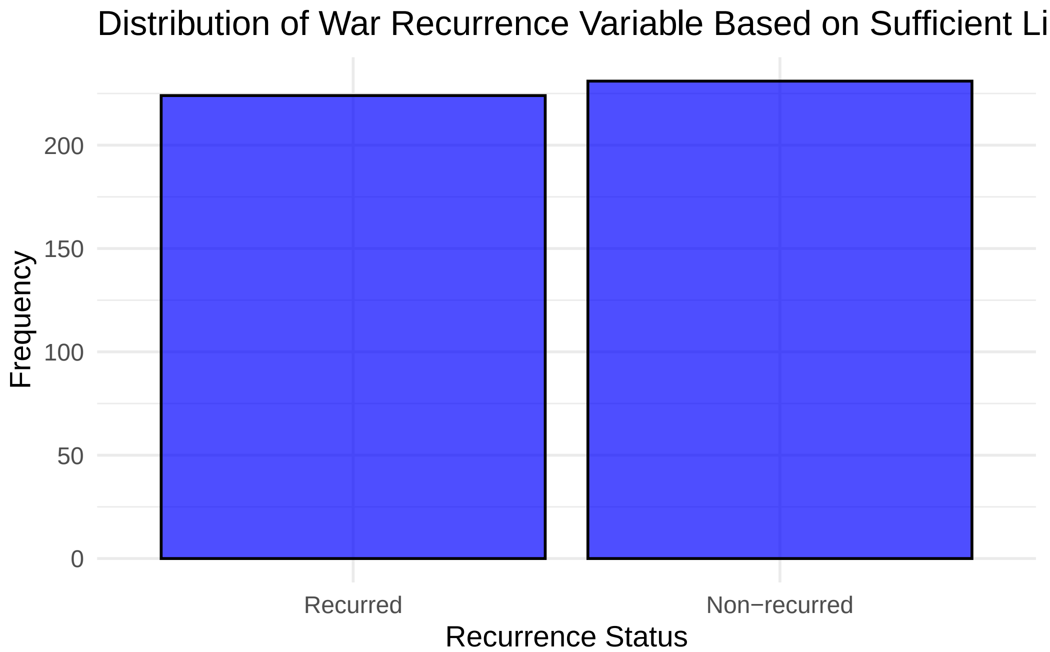
```
# A tibble: 455 x 125
   episode_id cname        year cease outcome country_name conflict_id start_year
        <dbl> <chr>       <dbl> <dbl>   <dbl> <chr>              <dbl>      <dbl>
 1          1 Bolivia (~   1947     0       4 Bolivia              200       1946
 2          2 Bolivia (~   1950     0       3 Bolivia              200       1949
```

```
3           3 Bolivia (~  1953     0        4 Bolivia                200       1952
4           4 Bolivia (~  1968     0        3 Bolivia                200       1967
5           5 China       1950     0        4 China                  202       1946
6           6 Greece      1950     0        3 Greece                 203       1946
7           7 Iran (Isl~  1947     0        3 Iran                   205       1946
8           8 Iran (Isl~  1969     0        5 Iran                   205       1966
9           9 Iran (Isl~  1989     0        5 Iran                   205       1979
10         10 Iran (Isl~  1991     0        5 Iran                   205       1990
# i 445 more rows
# i 117 more variables: end_year <dbl>, peace_cease <dbl>,
#   peace_cease_notes <chr>, wdi_expmilge <dbl>, wdi_armimp <dbl>,
#   wdi_expmil <dbl>, bicc_hw <dbl>, bicc_gmi <dbl>, atop_ally <dbl>,
#   atop_number <dbl>, bicc_milexp <dbl>, fe_etfra <dbl>, al_ethnic2000 <dbl>,
#   al_language2000 <dbl>, al_religion2000 <dbl>, fe_cultdiv <dbl>,
#   p_durable <dbl>, wdi_gdpcapcur <dbl>, p_polity2 <dbl>, pko_u <dbl>, ...
```

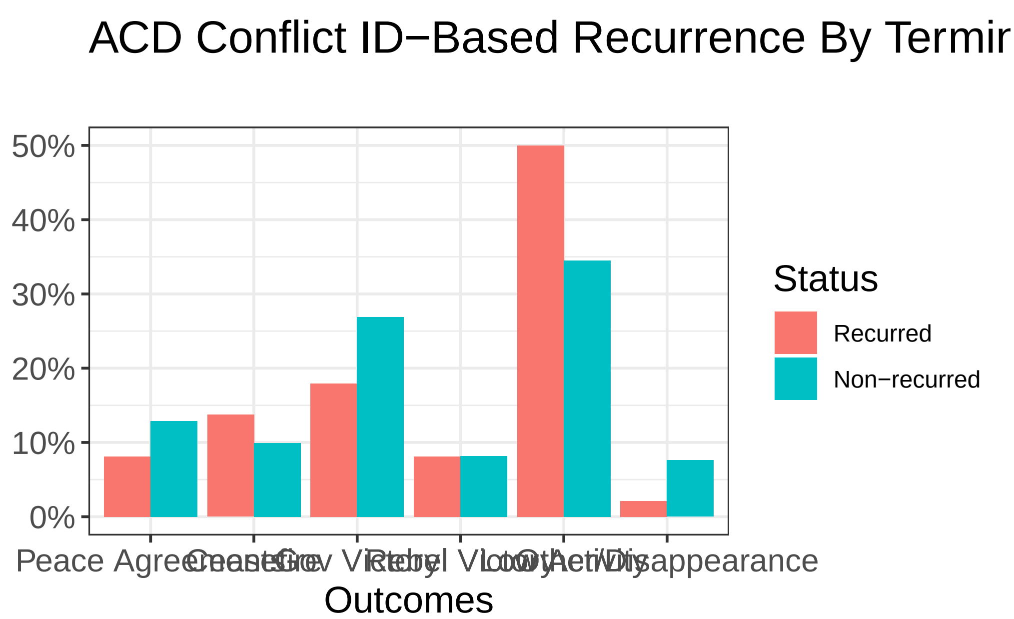
```r
# Recode the factor levels
episodes$recur0 <- recode_factor(episodes$recur_any, '1'="Recurred", '0'="Non-recurred")

# Plot the bar chart
ggplot(episodes, aes(x=recur0)) +
  geom_bar(fill="blue", color="black", alpha=0.7) +
  labs(title="Distribution of War Recurrence Variable Based on ACD Conflict ID",
       x="Recurrence Status",
       y="Frequency") +
  theme_minimal()
```




```
# Filter out rows with NA in the recurr column
episodes <- episodes %>% filter(!is.na(recur_side))

# Recode the factor levels
episodes$recur1 <- recode_factor(episodes$recur_side, '1'="Recurred", '0'="Non-recurred")

# Plot the bar chart
ggplot(episodes, aes(x=recur1)) +
  geom_bar(fill="blue", color="black", alpha=0.7) +
  labs(title="Distribution of War Recurrence Variable Based on Sufficient Linkage",
       x="Recurrence Status",
       y="Frequency") +
  theme_minimal()
```

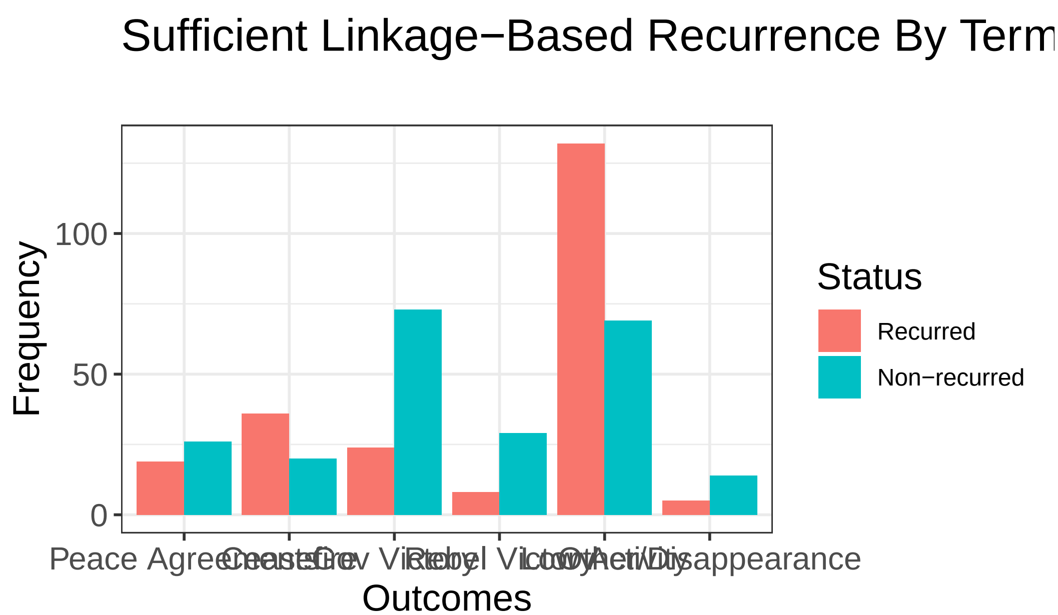



```
recur_outcome<- table(episodes$recur0, episodes$outcome)
recur_outcome_p<- prop.table(recur_outcome, 1)
recur_outcome_df <- as.data.frame(recur_outcome_p)
names(recur_outcome_df) <- c("recur0", "outcome",  "Frequency")
recur_outcome_df$outcome<-recode_factor(recur_outcome_df$outcome, '1'="Peace Agreements", '2
recur_by_outcome<-ggplot(recur_outcome_df, aes(x=outcome, y=Frequency, fill=recur0)) + geom_
  scale_y_continuous(label=percent) +
  labs(title="ACD Conflict ID-Based Recurrence By Termination Outcomes",
    caption="",
    subtitle="",
    x="Outcomes", y="", fill="Status")+
  theme_bw() +
  theme(title=element_text(size=14), axis.text=element_text(size=12))
recur_by_outcome
```



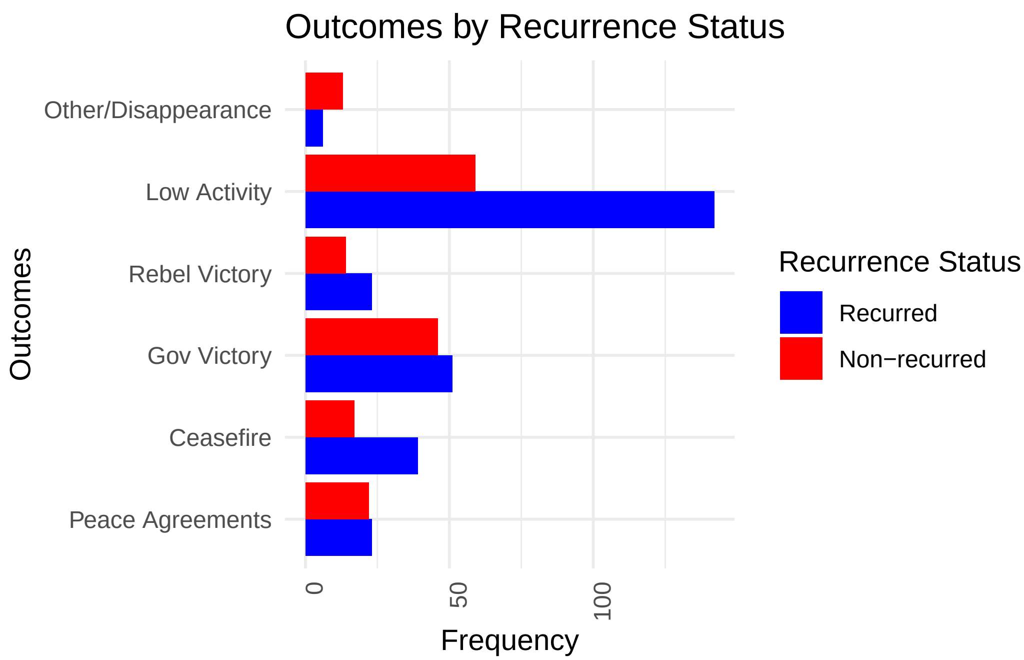

```r
# Create a frequency table
recur_outcome <- table(episodes$recur1, episodes$outcome)

# Convert the table to a data frame for ggplot
recur_outcome_df <- as.data.frame(recur_outcome)
names(recur_outcome_df) <- c("recur1", "outcome", "Frequency")

# Recode the outcome factor with descriptive names
recur_outcome_df$outcome <- recode_factor(recur_outcome_df$outcome,
                                          '1' = "Peace Agreements",
                                          '2' = "Ceasefire",
                                          '3' = "Gov Victory",
                                          '4' = "Rebel Victory",
                                          '5' = "Low Activity",
                                          '6' = "Other/Disappearance")

# Plot the bar chart using frequencies
recur_by_outcome <- ggplot(recur_outcome_df, aes(x = outcome, y = Frequency, fill = recur1))
  geom_col(position = "dodge") +
  labs(title = "Sufficient Linkage-Based Recurrence By Termination Outcomes",
       caption = "",
       subtitle = "",
       x = "Outcomes", y = "Frequency", fill = "Status") +
```

```
  theme_bw() +
  theme(title = element_text(size = 14), axis.text = element_text(size = 12))

recur_by_outcome
```



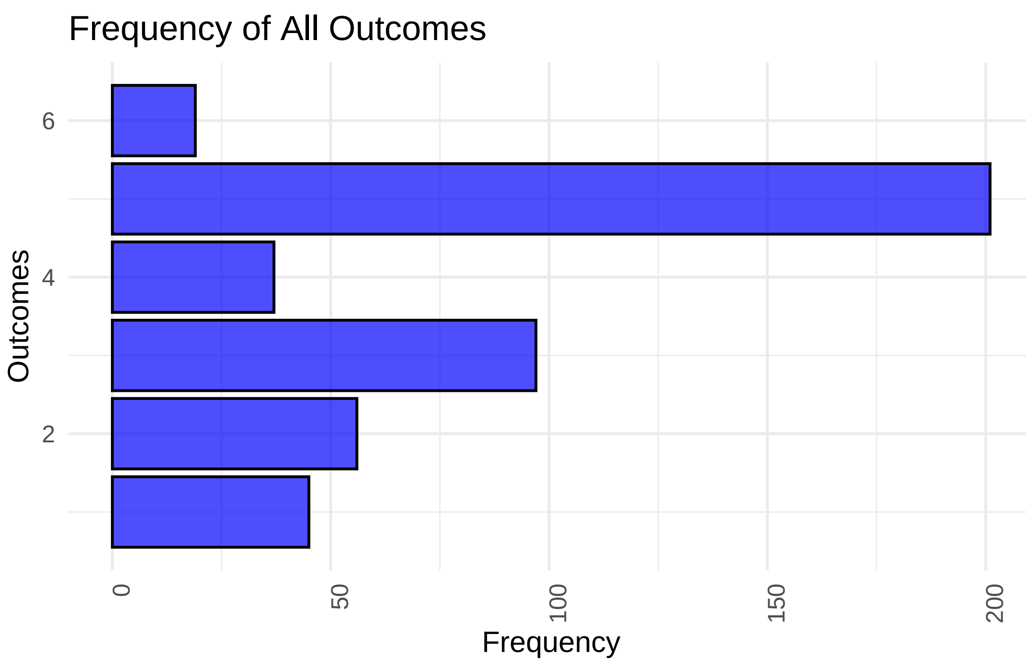

```
# Recode the outcome variable for better readability
episodes$outcome_label <- recode_factor(episodes$outcome,
                                  '1' = "Peace Agreements",
                                  '2' = "Ceasefire",
                                  '3' = "Gov Victory",
                                  '4' = "Rebel Victory",
                                  '5' = "Low Activity",
                                  '6' = "Other/Disappearance")

# Bar chart of the outcome variable separated by recur0
outcome_by_recur <- ggplot(episodes, aes(x=outcome_label, fill=factor(recur0))) +
  geom_bar(position="dodge") +
  labs(title="Outcomes by Recurrence Status",
       x="Outcomes",
       y="Frequency",
       fill="Recurrence Status") +
  scale_fill_manual(values=c("blue", "red")) + # Use manual colors if needed
```

```
  theme_minimal() +
  coord_flip()+
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) # Rotate x-axis labels for better
outcome_by_recur
```

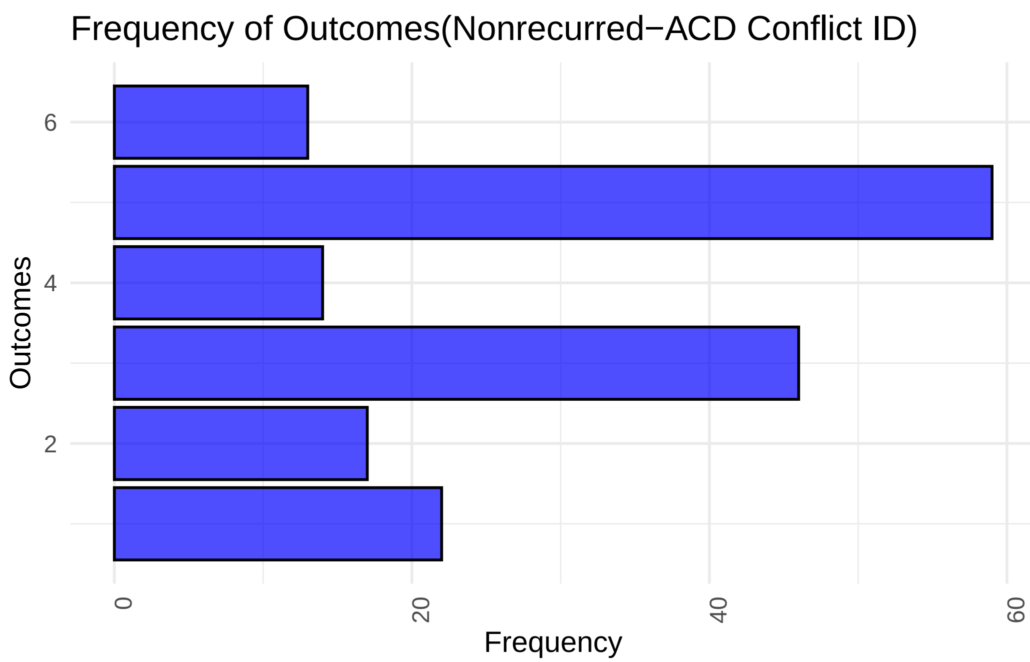


Cross-Tab Viz1

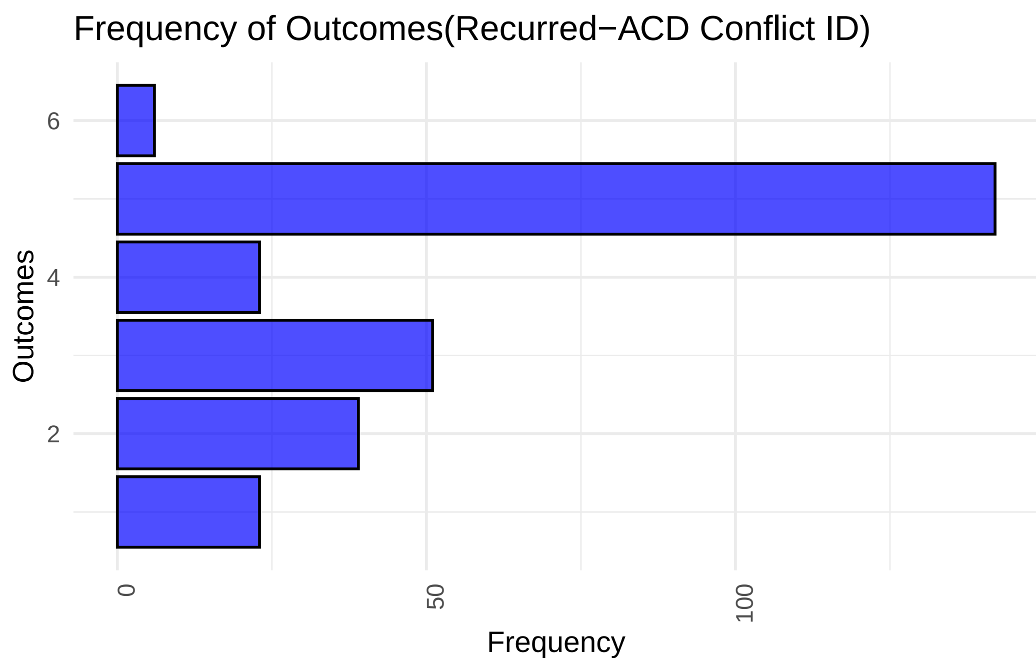
```
outcome_freq <- ggplot(episodes, aes(x=outcome)) +
  geom_bar(fill="blue", color="black", alpha=0.7) +
  labs(title="Frequency of All Outcomes",
       x="Outcomes",
       y="Frequency") +
  theme_minimal() +
  coord_flip()+
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) # Rotate x-axis labels for better
outcome_freq
```



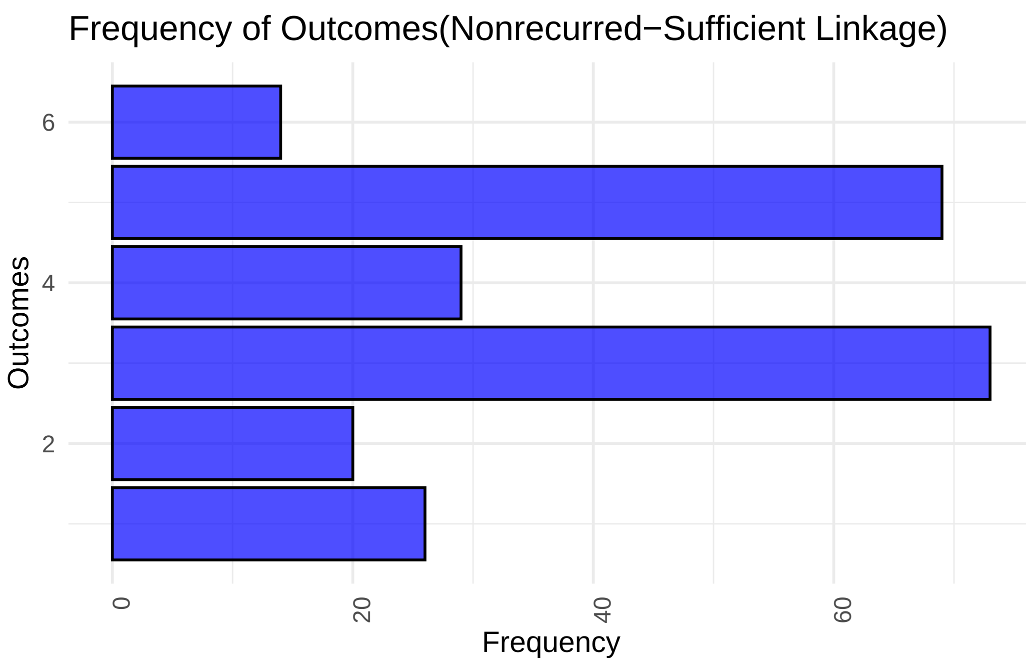

```
# Bar chart of the outcome variable without considering recur0
recur00_data<-subset(episodes, recur0 =="Non-recurred")
outcome_freq <- ggplot(recur00_data, aes(x=outcome)) +
  geom_bar(fill="blue", color="black", alpha=0.7) +
  labs(title="Frequency of Outcomes(Nonrecurred-ACD Conflict ID)",
       x="Outcomes",
       y="Frequency") +
  theme_minimal() +
  coord_flip()+
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) # Rotate x-axis labels for better
outcome_freq
```



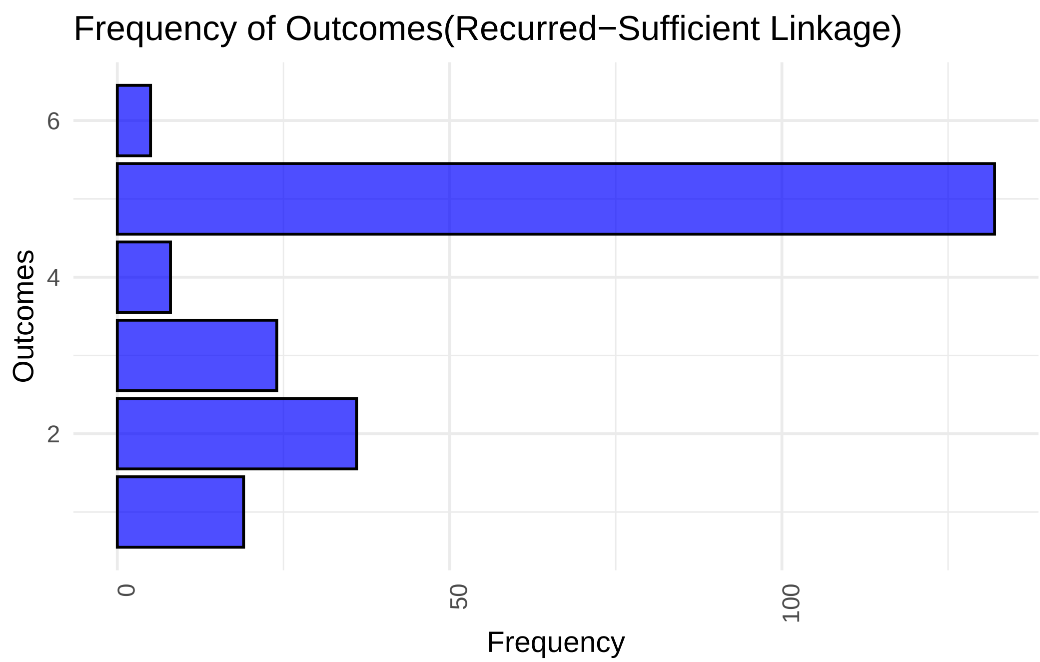

```
# Bar chart of the outcome variable without considering recur0
recur01_data<-subset(episodes, recur0 =="Recurred")
outcome_freq <- ggplot(recur01_data, aes(x=outcome)) +
  geom_bar(fill="blue", color="black", alpha=0.7) +
  labs(title="Frequency of Outcomes(Recurred-ACD Conflict ID)",
       x="Outcomes",
       y="Frequency") +
  theme_minimal() +
  coord_flip()+
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) # Rotate x-axis labels for better
outcome_freq
```




```
# Bar chart of the outcome variable without considering recur0
recur10_data<-subset(episodes, recur1 =="Non-recurred")
outcome_freq <- ggplot(recur10_data, aes(x=outcome)) +
  geom_bar(fill="blue", color="black", alpha=0.7) +
  labs(title="Frequency of Outcomes(Nonrecurred-Sufficient Linkage)",
       x="Outcomes",
       y="Frequency") +
  theme_minimal() +
  coord_flip()+
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) # Rotate x-axis labels for better
outcome_freq
```

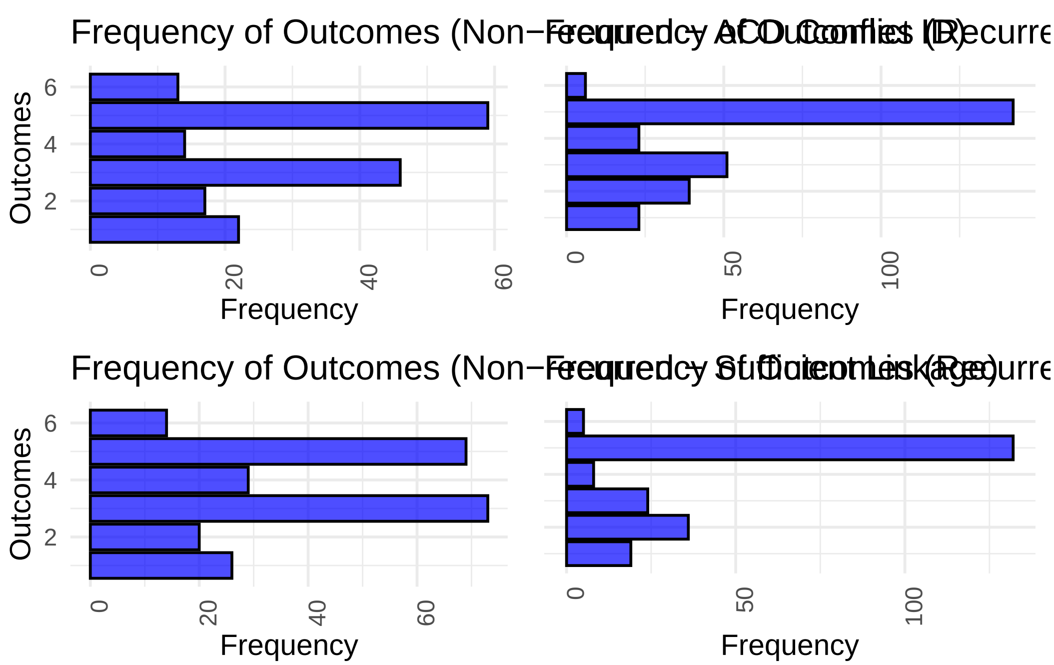



```r
# Bar chart of the outcome variable without considering recur0
recur11_data<-subset(episodes, recur1 =="Recurred")
outcome_freq <- ggplot(recur11_data, aes(x=outcome)) +
  geom_bar(fill="blue", color="black", alpha=0.7) +
  labs(title="Frequency of Outcomes(Recurred-Sufficient Linkage)",
       x="Outcomes",
       y="Frequency") +
  theme_minimal() +
  coord_flip()+
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) # Rotate x-axis labels for better
outcome_freq
```




```
# Prepare the first plot (Non-recurred - ACD Conflict ID)
recur00_data <- subset(episodes, recur0 == "Non-recurred")
outcome_freq00 <- ggplot(recur00_data, aes(x=outcome)) +
  geom_bar(fill="blue", color="black", alpha=0.7) +
  labs(title="Frequency of Outcomes (Non-recurred - ACD Conflict ID)",
       x="Outcomes", y="Frequency") +
  theme_minimal() + coord_flip() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))

# Prepare the second plot (Recurred - ACD Conflict ID)
recur01_data <- subset(episodes, recur0 == "Recurred")
outcome_freq01 <- ggplot(recur01_data, aes(x=outcome)) +
  geom_bar(fill="blue", color="black", alpha=0.7) +
  labs(title="Frequency of Outcomes (Recurred - ACD Conflict ID)",
       x="Outcomes", y="Frequency") +
  theme_minimal() + coord_flip() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1), axis.text.y = element_blank(), ax

# Prepare the third plot (Non-recurred - Sufficient Linkage)
recur10_data <- subset(episodes, recur1 == "Non-recurred")
outcome_freq10 <- ggplot(recur10_data, aes(x=outcome)) +
  geom_bar(fill="blue", color="black", alpha=0.7) +
  labs(title="Frequency of Outcomes (Non-recurred - Sufficient Linkage)",
```

```
       x="Outcomes", y="Frequency") +
  theme_minimal() + coord_flip() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))

# Prepare the fourth plot (Recurred - Sufficient Linkage)
recur11_data <- subset(episodes, recur1 == "Recurred")
outcome_freq11 <- ggplot(recur11_data, aes(x=outcome)) +
  geom_bar(fill="blue", color="black", alpha=0.7) +
  labs(title="Frequency of Outcomes (Recurred - Sufficient Linkage)",
       x="Outcomes", y="Frequency") +
  theme_minimal() + coord_flip() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1), axis.text.y = element_blank(), ax

# Combine all four plots into one
combined_plot <- grid.arrange(outcome_freq00, outcome_freq01, outcome_freq10, outcome_freq11
```



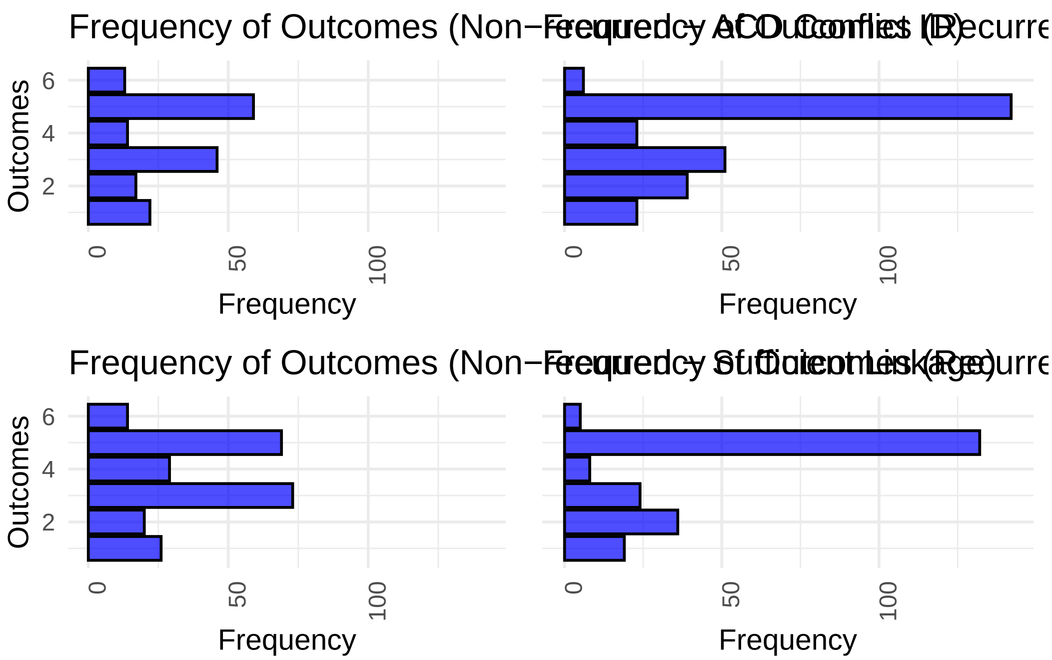

```
print(combined_plot)
```

```
TableGrob (2 x 2) "arrange": 4 grobs
  z     cells    name           grob
1 1 (1-1,1-1) arrange gtable[layout]
2 2 (1-1,2-2) arrange gtable[layout]
```

```
3 3 (2-2,1-1) arrange gtable[layout]
4 4 (2-2,2-2) arrange gtable[layout]
```

```r
# Calculate the maximum frequency across all subsets
max_freq <- max(
  table(episodes$recur0, episodes$outcome),
  table(episodes$recur1, episodes$outcome)
)

# Prepare the first plot (Non-recurred - ACD Conflict ID)
recur00_data <- subset(episodes, recur0 == "Non-recurred")
outcome_freq00 <- ggplot(recur00_data, aes(x=outcome)) +
  geom_bar(fill="blue", color="black", alpha=0.7) +
  labs(title="Frequency of Outcomes (Non-recurred - ACD Conflict ID)",
       x="Outcomes", y="Frequency") +
  theme_minimal() +
  coord_flip() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylim(c(0, max_freq))

# Prepare the second plot (Recurred - ACD Conflict ID)
recur01_data <- subset(episodes, recur0 == "Recurred")
outcome_freq01 <- ggplot(recur01_data, aes(x=outcome)) +
  geom_bar(fill="blue", color="black", alpha=0.7) +
  labs(title="Frequency of Outcomes (Recurred - ACD Conflict ID)",
       x="Outcomes", y="Frequency") +
  theme_minimal() +
  coord_flip() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1), axis.text.y = element_blank(), ax
  ylim(c(0, max_freq))

# Prepare the third plot (Non-recurred - Sufficient Linkage)
recur10_data <- subset(episodes, recur1 == "Non-recurred")
outcome_freq10 <- ggplot(recur10_data, aes(x=outcome)) +
  geom_bar(fill="blue", color="black", alpha=0.7) +
  labs(title="Frequency of Outcomes (Non-recurred - Sufficient Linkage)",
       x="Outcomes", y="Frequency") +
  theme_minimal() +
  coord_flip() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylim(c(0, max_freq))
```

```
# Prepare the fourth plot (Recurred - Sufficient Linkage)
recur11_data <- subset(episodes, recur1 == "Recurred")
outcome_freq11 <- ggplot(recur11_data, aes(x=outcome)) +
  geom_bar(fill="blue", color="black", alpha=0.7) +
  labs(title="Frequency of Outcomes (Recurred - Sufficient Linkage)",
       x="Outcomes", y="Frequency") +
  theme_minimal() +
  coord_flip() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1), axis.text.y = element_blank(), axi
  ylim(c(0, max_freq))

# Combine all four plots into one
combined_plot <- grid.arrange(outcome_freq00, outcome_freq01, outcome_freq10, outcome_freq11
```




```
print(combined_plot)
```

```
TableGrob (2 x 2) "arrange": 4 grobs
  z     cells    name           grob
1 1 (1-1,1-1) arrange gtable[layout]
2 2 (1-1,2-2) arrange gtable[layout]
3 3 (2-2,1-1) arrange gtable[layout]
4 4 (2-2,2-2) arrange gtable[layout]
```

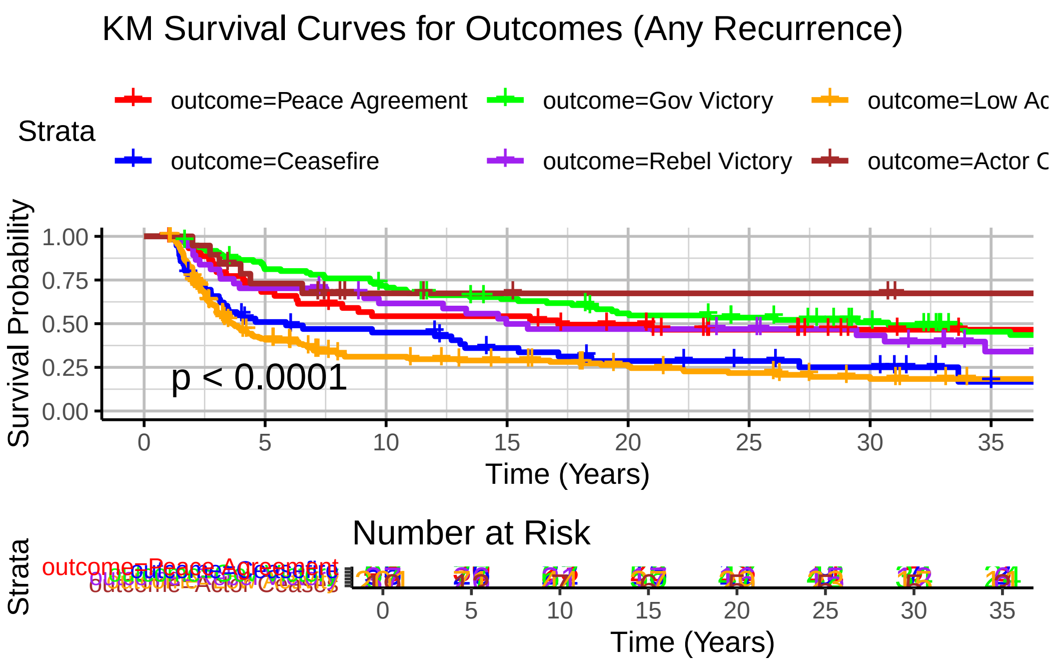
Kaplan-Meier survival curves

```
# Create a new dataframe from episodes

new_episodes <- episodes
new_episodes<-new_episodes[, c("outcome", "recur_side", "recur_any", "time_to_recur", "time_t
new_episodes<-na.omit(new_episodes)


#Convert the numerical outcome variable to a factor with appropriate labels
new_episodes$outcome <- factor(new_episodes$outcome, levels = c(1, 2, 3, 4, 5, 6),
                               labels = c("Peace Agreement", "Ceasefire", "Gov Victory", "Reb

# Create new variables for time in years
new_episodes$time_to_recur_any_years <- new_episodes$time_to_recur_any / 365.25
new_episodes$time_to_recur_years <- new_episodes$time_to_recur / 365.25

# Trim follow-up time to 20,000 days (approximately 54.8 years), and create corresponding var
new_episodes$time_to_recur_any_years <- pmin(new_episodes$time_to_recur_any_years, 20000 / 36
new_episodes$time_to_recur_years <- pmin(new_episodes$time_to_recur_years, 20000 / 365.25)

# Fit the Kaplan-Meier survival curves for first plot (Any Recurrence)
km_fit_outcome_any <- survfit(Surv(time_to_recur_any_years, recur_any) ~ outcome, data = new_

# Plot the Kaplan-Meier survival curves with percentages in the risk table (Any Recurrence)
km_plot_any <- ggsurvplot(km_fit_outcome_any, data = new_episodes,
                          pval = TRUE, conf.int = FALSE,
                          risk.table = TRUE, risk.table.col = "strata",
                          risk.table.y.text.col = TRUE,
                          risk.table.height = 0.25,
                          risk.table.title = "Number at Risk",
                          xlim = c(0, 35),  # Limit x-axis to 35 years
                          break.time.by = 5,  # Break x-axis every 5 years
                          ggtheme = theme_classic() +  # Use classic theme
                                    theme(panel.grid.major = element_line(color = "gray", siz
                                          panel.grid.minor = element_line(color = "lightgray"
                          palette = c("red", "blue", "green", "purple", "orange", "brown"),
                          title = "KM Survival Curves for Outcomes (Any Recurrence)",
                          xlab = "Time (Years)", ylab = "Survival Probability",
                          tables.theme = theme_classic())  # Use classic theme for the risk t
```

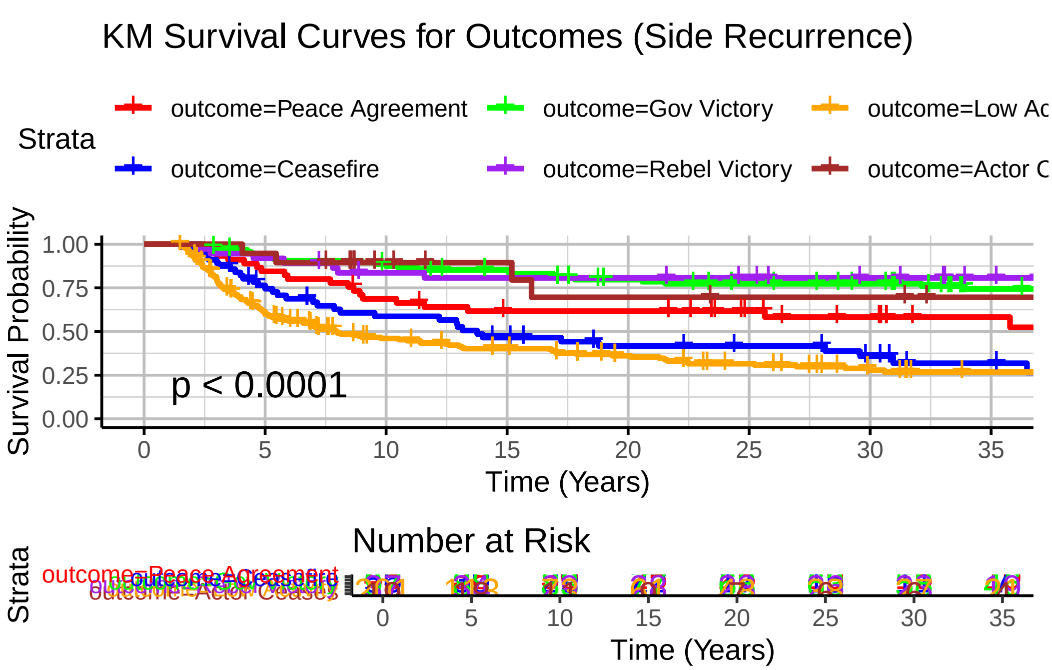
```
Warning: The `size` argument of `element_line()` is deprecated as of ggplot2 3.4.0.
```

```
i Please use the `linewidth` argument instead.
```

```r
# Fit the Kaplan-Meier survival curves for the second plot (Side Recurrence)
km_fit_outcome_side <- survfit(Surv(time_to_recur_years, recur_side) ~ outcome, data = new_ep

# Plot the Kaplan-Meier survival curves with percentages in the risk table for the second plo
km_plot_side <- ggsurvplot(km_fit_outcome_side, data = new_episodes,
                           pval = TRUE, conf.int = FALSE,
                           risk.table = TRUE, risk.table.col = "strata",
                           risk.table.y.text.col = TRUE,
                           risk.table.height = 0.25,
                           risk.table.title = "Number at Risk",
                           xlim = c(0, 35),  # Limit x-axis to 35 years
                           break.time.by = 5,  # Break x-axis every 5 years
                           ggtheme = theme_classic() +  # Use classic theme
                                     theme(panel.grid.major = element_line(color = "gray", si
                                           panel.grid.minor = element_line(color = "lightgray
                           palette = c("red", "blue", "green", "purple", "orange", "brown"),
                           title = "KM Survival Curves for Outcomes (Side Recurrence)",
                           xlab = "Time (Years)", ylab = "Survival Probability",
                           tables.theme = theme_classic())  # Use classic theme for the risk

# Print plots
print(km_plot_any)
```




```
print(km_plot_side)
```



Cox Models

Cox P Models

```
m1<- coxph(Surv(time_to_recur_any, recur_any) ~dis+cease+govvic+rebvic+lowac, data = episodes

m2<- coxph(Surv(time_to_recur_any, recur_any)~dis+cease+govvic+rebvic+lowac+pko_u+log_dur+col

m3<- coxph(Surv(time_to_recur, recur_side) ~dis+cease+govvic+rebvic+lowac, data = episodes, c

m4<- coxph(Surv(time_to_recur, recur_side)~dis+cease+govvic+rebvic+lowac+pko_u+log_dur+cold_w

#stargazer(m1, m2, m3, m4, type = "text")

stargazer(m1, m2, m3, m4,
          ord.intercepts = TRUE,
          dep.var.labels = c("UCDP ID-Based War Recurrence", "UCDP ID-Based War Recurrence",
          covariate.labels = c("Actor Ceases", "Ceasefire", "Government Victory", "Rebel Vic
```

```
==========================================================================================
                                          Dependent variable:
                           ---------------------------------------------------------------
                                UCDP ID-Based War Recurrence           UCDP ID-Based War Rec
                                  (1)                 (2)                 (3)
------------------------------------------------------------------------------------------
Actor Ceases                     -0.484              0.169               -0.447
                                (0.459)             (0.606)             (0.503)

Ceasefire                       0.603**             1.182***            0.658**           (
                                (0.263)             (0.366)             (0.284)

Government Victory               -0.184              0.283              -0.757**
                                (0.252)             (0.377)             (0.308)

Rebel Victory                    0.105               0.822              -0.894**
                                (0.295)             (0.444)             (0.422)
```

Low Activity	0.783***	1.186***	0.884***
	(0.226)	(0.335)	(0.246)
Peacekeeping Missions		-0.585**	
		(0.282)	
Log(Duration)		0.059*	
		(0.029)	
Cold War		-0.492***	
		(0.175)	
Log(GDP per Capita)		-0.080	
		(0.057)	
Number of Veto Players		0.236***	
		(0.088)	
War over Government		-1.431***	
		(1.029)	
War over Territory		-1.321***	
		(1.031)	
Polity Score		-0.015	
		(0.022)	
Ethnic Fractionalization		1.508**	
		(0.628)	
Coalition Size		0.028	
		(0.055)	
Log(Population)		-0.106	
		(0.365)	
Language Fractionalization		-1.130	
		(0.669)	
Power-Sharing		-0.192	
		(0.686)	

```
Religion Fractionalization                                 0.190
                                                          (0.247)

----------------------------------------------------------------------------------------------
Observations                         455                  329                  455
R2                                  0.107                0.209                0.197
Max. Possible R2                    0.999                0.999                0.996
Log Likelihood                   -1,550.938           -1,114.252           -1,219.034
Wald Test                   48.250*** (df = 5) 264.530*** (df = 19) 94.070*** (df = 5) 310.54
LR Test                     51.640*** (df = 5) 76.996*** (df = 19)  99.780*** (df = 5) 115.99
Score (Logrank) Test        51.438*** (df = 5) 78.548*** (df = 19)  95.665*** (df = 5) 118.77
==============================================================================================
Note:                                                                              *p<0.1; **p<0
```

```r
#al_language2000
#al_religion2000
#fe_etfra
#fe_cultdiv
#al_ethnic2000






# Check proportional hazards assumption
m1_t<- cox.zph(m1)
m2_t<- cox.zph(m2)
m3_t<- cox.zph(m3)
m4_t<- cox.zph(m4)
print(m1_t)
```

```
         chisq df       p
dis     0.0994  1 0.75260
cease   1.1805  1 0.27725
govvic 10.9580  1 0.00093
rebvic  1.1021  1 0.29381
lowac   6.0333  1 0.01404
GLOBAL 14.2584  5 0.01405
```

```r
print(m2_t)
```

```
                     chisq df     p
```

```
dis               0.0444332  1 0.833
cease             0.1483013  1 0.700
govvic            3.5989805  1 0.058
rebvic            0.0005874  1 0.981
lowac             0.8331863  1 0.361
pko_u             1.2850393  1 0.257
log_dur           5.4597689  1 0.019
cold_war          0.0324631  1 0.857
log_gdp           2.7998279  1 0.094
veto_u            1.6101650  1 0.204
ter_war           0.0000616  1 0.994
gov_war           0.0010633  1 0.974
p_polity2         4.4359281  1 0.035
fe_etfra          0.0239740  1 0.877
log_pop           0.0839223  1 0.772
al_religion2000   0.0111114  1 0.916
fe_cultdiv        1.2799303  1 0.258
W4                2.7115601  1 0.100
ps_original       3.4820684  1 0.062
GLOBAL           23.3819547 19 0.221
```

```r
print(m3_t)
```

```
       chisq df     p
dis    2.125  1 0.145
cease  3.704  1 0.054
govvic 0.645  1 0.422
rebvic 0.602  1 0.438
lowac  4.589  1 0.032
GLOBAL 8.251  5 0.143
```

```r
print(m4_t)
```

```
                      chisq df          p
dis              2.69519262  1     0.1007
cease            4.56012230  1     0.0327
govvic           0.22149714  1     0.6379
rebvic           3.51213317  1     0.0609
lowac            2.44894340  1     0.1176
pko_u            0.01881588  1     0.8909
log_dur         25.99137111  1 0.00000034
```

```
cold_war         6.78720120  1     0.0092
log_gdp          0.04147132  1     0.8386
veto_u           5.16563745  1     0.0230
ter_war          0.79338533  1     0.3731
gov_war          0.84177544  1     0.3589
p_polity2        0.00027204  1     0.9868
fe_etfra         0.00000377  1     0.9985
log_pop          0.84916823  1     0.3568
al_religion2000  0.05014842  1     0.8228
fe_cultdiv       0.05344662  1     0.8172
W4               0.16978739  1     0.6803
ps_original      0.05276456  1     0.8183
GLOBAL          42.42831304 19     0.0016
```

```r
#vif(m1)
#vif(m2)
#vif(m3)
#vif(m4)







library(corrplot)
```

```
corrplot 0.92 loaded
```

```r
columns_of_interest <- episodes[, c("al_language2000", "al_religion2000", "fe_etfra", "fe_cul
cor(columns_of_interest, use = "complete.obs")
```

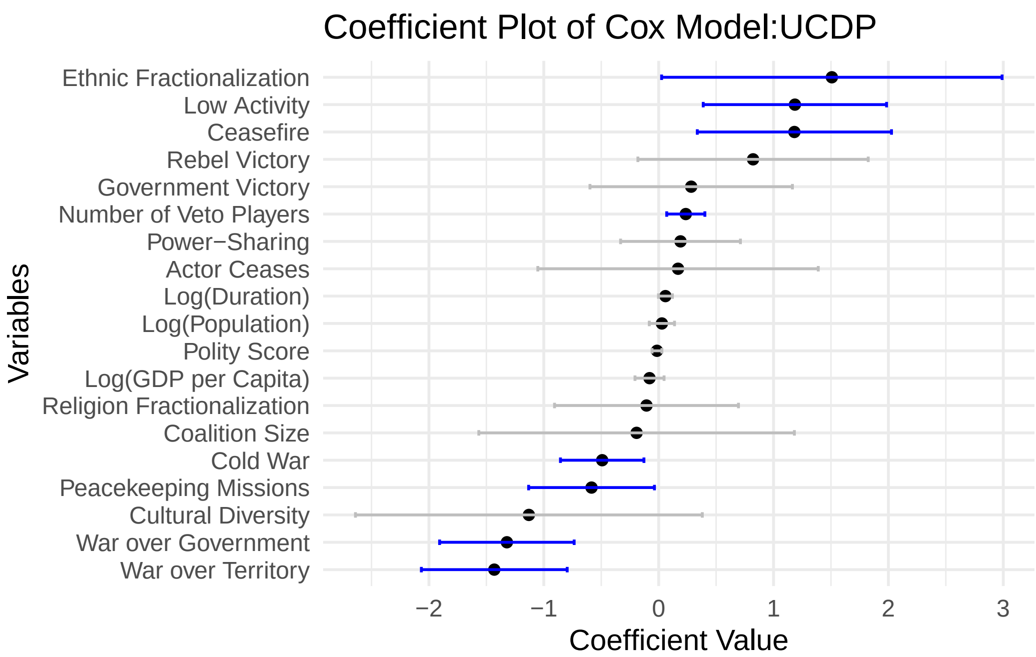
```
                al_language2000 al_religion2000  fe_etfra fe_cultdiv
al_language2000       1.0000000       0.2920441 0.7176872  0.7396354
al_religion2000       0.2920441       1.0000000 0.3563721  0.1479685
fe_etfra              0.7176872       0.3563721 1.0000000  0.7723940
fe_cultdiv            0.7396354       0.1479685 0.7723940  1.0000000
```

Coefficent Plots from Cox

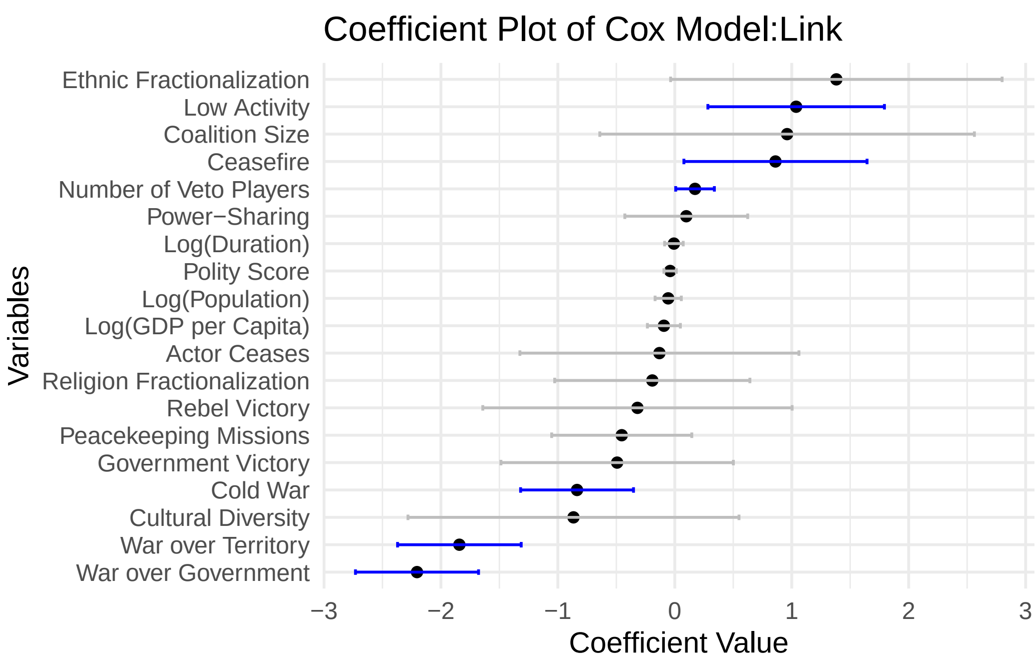
```r
# Extracting model coefficients and robust standard errors from the summary
coef_summary <- summary(m2)$coefficients

# Rename terms for better readability
terms_rename <- c(
  "dis" = "Actor Ceases",
  "cease" = "Ceasefire",
  "govvic" = "Government Victory",
  "rebvic" = "Rebel Victory",
  "lowac" = "Low Activity",
  "pko_u" = "Peacekeeping Missions",
  "log_dur" = "Log(Duration)",
  "cold_war" = "Cold War",
  "log_gdp" = "Log(GDP per Capita)",
  "veto_u" = "Number of Veto Players",
  "gov_war" = "War over Government",
  "ter_war" = "War over Territory",
  "p_polity2" = "Polity Score",
  "fe_etfra" = "Ethnic Fractionalization",
  "W4" = "Coalition Size",
  "log_pop" = "Log(Population)",
  "ps_original" = "Power-Sharing",
  "al_religion2000" = "Religion Fractionalization",
  "fe_cultdiv" = "Cultural Diversity"
)

# Create a dataframe for plotting
df_coef <- data.frame(
  Term = rownames(coef_summary),
  Estimate = coef_summary[, "coef"],
  StdErr = coef_summary[, "robust se"],
  Lower = coef_summary[, "coef"] - 1.96 * coef_summary[, "robust se"],
  Upper = coef_summary[, "coef"] + 1.96 * coef_summary[, "robust se"],
  p_value = coef_summary[, "Pr(>|z|)"]
)

# Apply the renaming to the Term column
df_coef$Term <- terms_rename[df_coef$Term]

# Order by Estimate from most negative to most positive
df_coef <- df_coef[order(df_coef$Estimate), ]
```

```r
# Plotting using ggplot2
library(ggplot2)
coef_ucdp<-ggplot(df_coef, aes(x = Estimate, y = reorder(Term, Estimate))) +
  geom_point() +
  geom_errorbarh(aes(xmin = Lower, xmax = Upper), height = 0.2, color = ifelse(df_coef$p_valu
  theme_minimal() +
  labs(title = "Coefficient Plot of Cox Model:UCDP",
       x = "Coefficient Value",
       y = "Variables")


# Extracting model coefficients and robust standard errors from the summary
coef_summary <- summary(m4)$coefficients

# Rename terms for better readability
terms_rename <- c(
  "dis" = "Actor Ceases",
  "cease" = "Ceasefire",
  "govvic" = "Government Victory",
  "rebvic" = "Rebel Victory",
  "lowac" = "Low Activity",
  "pko_u" = "Peacekeeping Missions",
  "log_dur" = "Log(Duration)",
  "cold_war" = "Cold War",
  "log_gdp" = "Log(GDP per Capita)",
  "veto_u" = "Number of Veto Players",
  "gov_war" = "War over Government",
  "ter_war" = "War over Territory",
  "p_polity2" = "Polity Score",
  "fe_etfra" = "Ethnic Fractionalization",
  "W4" = "Coalition Size",
  "log_pop" = "Log(Population)",
  "ps_original" = "Power-Sharing",
  "al_religion2000" = "Religion Fractionalization",
  "fe_cultdiv" = "Cultural Diversity"
)

# Create a dataframe for plotting
df_coef <- data.frame(
  Term = rownames(coef_summary),
  Estimate = coef_summary[, "coef"],
  StdErr = coef_summary[, "robust se"],
```

```
  Lower = coef_summary[, "coef"] - 1.96 * coef_summary[, "robust se"],
  Upper = coef_summary[, "coef"] + 1.96 * coef_summary[, "robust se"],
  p_value = coef_summary[, "Pr(>|z|)"]
)

# Apply renaming to the Term column
df_coef$Term <- terms_rename[df_coef$Term]

# Order by Estimate from most negative to most positive
df_coef <- df_coef[order(df_coef$Estimate), ]

# Plotting using ggplot2
library(ggplot2)
coef_link<-ggplot(df_coef, aes(x = Estimate, y = reorder(Term, Estimate))) +
  geom_point() +
  geom_errorbarh(aes(xmin = Lower, xmax = Upper), height = 0.2, color = ifelse(df_coef$p_valu
  theme_minimal() +
  labs(title = "Coefficient Plot of Cox Model:Link",
       x = "Coefficient Value",
       y = "Variables")
coef_ucdp
```



```
coef_link
```



Survival Rates from Cox

```
library(survival)
library(dplyr)
library(survminer)
library(ggplot2)

# Define factor levels and labels
outcome_levels <- c("Peace Agreement", "Ceasefire", "Gov Victory", "Rebel Victory", "Low Acti

# Convert outcome variable to factor with specified levels in the original dataset
episodes$outcomes <- factor(episodes$outcome, levels = 1:6, labels = outcome_levels)

# Convert the time variables from days to years in the original dataset
episodes$time_to_recur_any_years <- episodes$time_to_recur_any / 365.25
episodes$time_to_recur_years <- episodes$time_to_recur / 365.25

# Fit Cox proportional hazards models with time in years
```

```r
m2 <- coxph(Surv(time_to_recur_any_years, recur_any) ~ as.factor(outcomes) + pko_u + log_dur

m4 <- coxph(Surv(time_to_recur_years, recur_side) ~ as.factor(outcomes) + pko_u + log_dur + c

# Create representative data without al_language2000
representative_data <- episodes %>%
  summarise(
    pko_u = median(as.numeric(pko_u), na.rm = TRUE),
    log_dur = mean(log_dur, na.rm = TRUE),
    cold_war = median(cold_war, na.rm = TRUE),
    log_gdp = mean(log_gdp, na.rm = TRUE),
    veto_u = median(veto_u, na.rm = TRUE),
    gov_war = median(gov_war, na.rm = TRUE),
    ter_war = median(ter_war, na.rm = TRUE),
    p_polity2 = mean(p_polity2, na.rm = TRUE),
    fe_etfra = mean(fe_etfra, na.rm = TRUE),
    W4 = mean(W4, na.rm = TRUE),
    log_pop = mean(log_pop, na.rm = TRUE),
    ps_original = median(ps_original, na.rm = TRUE),
    al_religion2000 = mean(al_religion2000, na.rm = TRUE),
    fe_cultdiv = mean(fe_cultdiv, na.rm = TRUE)
  )

# Expand the data to include different outcomes
expanded_data <- do.call(rbind, replicate(6, representative_data, simplify = FALSE))
expanded_data$outcomes <- factor(rep(outcome_levels, each = 1), levels = outcome_levels)

# Generate survival curves based on the Cox model for m2
surv_fits_any <- survfit(m2, newdata = expanded_data)

# Define colors for the plot
colors <- c("Peace Agreement" = "red", "Ceasefire" = "blue", "Gov Victory" = "green", "Rebel

# Plotting survival curves for Any Recurrence with x-axis in years
plot_any <- ggsurvplot(
  surv_fits_any,
  data = expanded_data,
  pval = TRUE,  # Show p-value for log-rank test
  conf.int = FALSE,  # Show confidence intervals
  xlab = "Time in Years",
  ylab = "Survival Probability",
  ggtheme = theme_bw(),
```

```r
  xlim = c(0, 30),  # Set the x-axis limit in years (adjust as needed)
  break.time.by = 5,  # Break x-axis every 5 years
  risk.table = TRUE,  # Show number at risk table
  surv.scale = "percent",
  legend.title = "Outcome",
  legend.labs = levels(expanded_data$outcomes),  # Automatically use factor labels
  title = "Survival Probability Over Time for War Recurrence Based on ACD Conflict ID",
  subtitle = "",
  caption = "",
  palette = colors,
  size = 1.2,  # Increase line width for better visibility
  linetype = "solid"  # Set line type to solid for all curves
)
```

```
Warning: `gather_()` was deprecated in tidyr 1.2.0.
i Please use `gather()` instead.
i The deprecated feature was likely used in the survminer package.
  Please report the issue at <https://github.com/kassambara/survminer/issues>.
```

```
Warning in .pvalue(fit, data = data, method = method, pval = pval, pval.coord = pval.coord,
 This is a null model.
```

```
Warning: No shared levels found between `names(values)` of the manual scale and the
data's fill values.
```

```
Warning in (function (survsummary, times, survtable = c("cumevents",
"risk.table", : The length of legend.labs should be 1
```

```r
# Customize the legend font size and axis titles for Any Recurrence plot
plot_any$plot <- plot_any$plot +
  theme(legend.text = element_text(size = 12),  # Increase legend text size
        legend.title = element_text(size = 14),  # Increase legend title size
        axis.title.x = element_text(size = 10),  # X-axis title size
        axis.title.y = element_text(size = 10),  # Y-axis title size
        axis.text.x = element_text(size = 10, angle = 45, hjust = 1),  # X-axis text size and
        axis.text.y = element_text(size = 10),   # Y-axis text size
        plot.title = element_text(size = 12))    # Plot title size

# Customize the risk table text for Any Recurrence plot
plot_any$table <- plot_any$table +
  scale_x_continuous(breaks = seq(0, 30, by = 5), labels = seq(0, 30, by = 5)) +  # Breaks ar
  theme(axis.text.x = element_text(size = 10, angle = 45, hjust = 1))  # Rotate risk table te
```

```
Scale for x is already present.
Adding another scale for x, which will replace the existing scale.
```

```
# Generate survival curves based on the Cox model for m4
surv_fits_side <- survfit(m4, newdata = expanded_data)

# Plotting survival curves for Side Recurrence with x-axis in years
plot_side <- ggsurvplot(
  surv_fits_side,
  data = expanded_data,
  pval = TRUE,  # Show p-value for log-rank test
  conf.int = FALSE,  # Show confidence intervals
  xlab = "Time in Years",
  ylab = "Survival Probability",
  ggtheme = theme_bw(),
  xlim = c(0, 30),  # Set the x-axis limit in years (adjust as needed)
  break.time.by = 5,  # Break x-axis every 5 years
  risk.table = TRUE,  # Show number at risk table
  surv.scale = "percent",
  legend.title = "Outcome",
  legend.labs = levels(expanded_data$outcomes),  # Automatically use factor labels
  title = "Survival Probability Over Time for War Recurrence Based on Sufficient Linkage",
  subtitle = "",
  caption = "",
  palette = colors,
  size = 1.2,  # Increase line width for better visibility
  linetype = "solid"  # Set line type to solid for all curves
)
```

```
Warning in .pvalue(fit, data = data, method = method, pval = pval, pval.coord = pval.coord,
 This is a null model.
```

```
Warning: No shared levels found between `names(values)` of the manual scale and the
data's fill values.
```

```
Warning in (function (survsummary, times, survtable = c("cumevents",
"risk.table", : The length of legend.labs should be 1
```

```
# Customize the legend font size and axis titles for Side Recurrence plot
plot_side$plot <- plot_side$plot +
  theme(legend.text = element_text(size = 12),  # Increase legend text size
```

```
        legend.title = element_text(size = 14),  # Increase legend title size
        axis.title.x = element_text(size = 10),  # X-axis title size
        axis.title.y = element_text(size = 10),  # Y-axis title size
        axis.text.x = element_text(size = 10, angle = 45, hjust = 1),  # X-axis text size and
        axis.text.y = element_text(size = 10),   # Y-axis text size
        plot.title = element_text(size = 12))    # Plot title size

# Customize the risk table text for Side Recurrence plot
plot_side$table <- plot_side$table +
  scale_x_continuous(breaks = seq(0, 30, by = 5), labels = seq(0, 30, by = 5)) +  # Breaks an
  theme(axis.text.x = element_text(size = 10, angle = 45, hjust = 1))  # Rotate risk table te
```

```
Scale for x is already present.
Adding another scale for x, which will replace the existing scale.
```

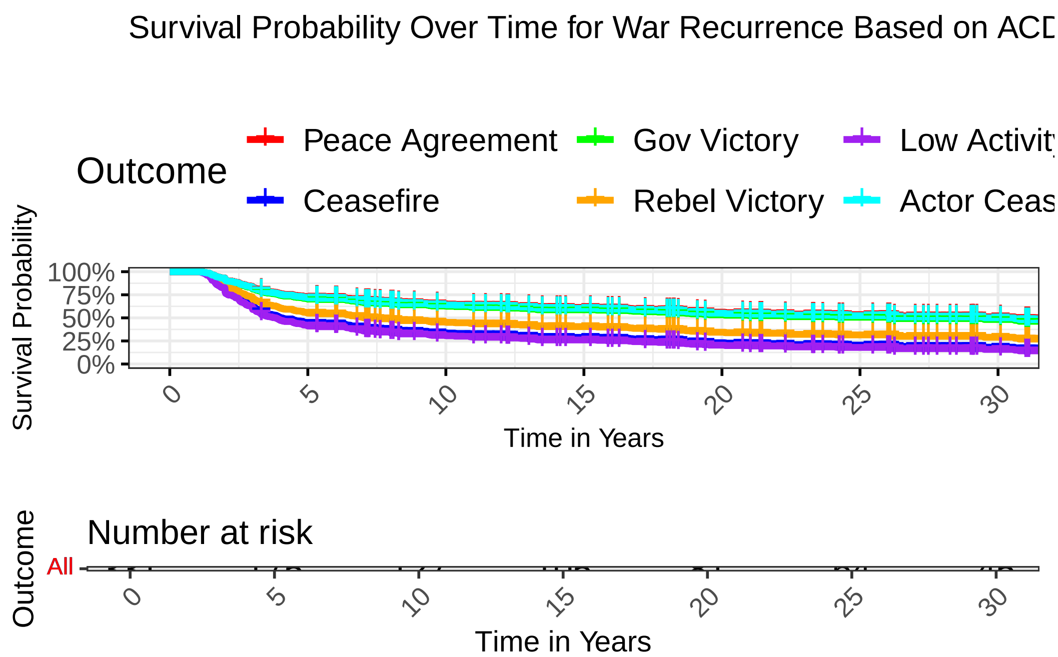
```
# Print the plots
print(plot_any)
```

```
Warning: No shared levels found between `names(values)` of the manual scale and the
data's fill values.
```

```
Warning: No shared levels found between `names(values)` of the manual scale and the
data's fill values.
No shared levels found between `names(values)` of the manual scale and the
data's fill values.
```

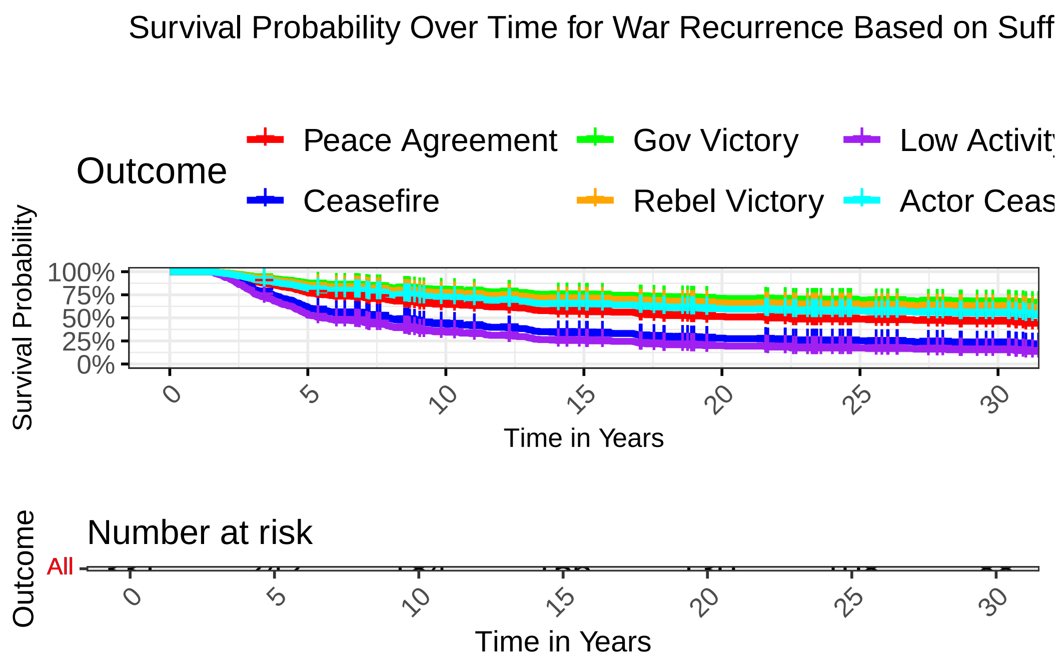
```
Warning: No shared levels found between `names(values)` of the manual scale and the
data's colour values.
```

```
Warning: No shared levels found between `names(values)` of the manual scale and the
data's fill values.
```




```
print(plot_side)
```

```
Warning: No shared levels found between `names(values)` of the manual scale and the
data's fill values.
No shared levels found between `names(values)` of the manual scale and the
data's fill values.
No shared levels found between `names(values)` of the manual scale and the
data's fill values.

Warning: No shared levels found between `names(values)` of the manual scale and the
data's colour values.

Warning: No shared levels found between `names(values)` of the manual scale and the
data's fill values.
```



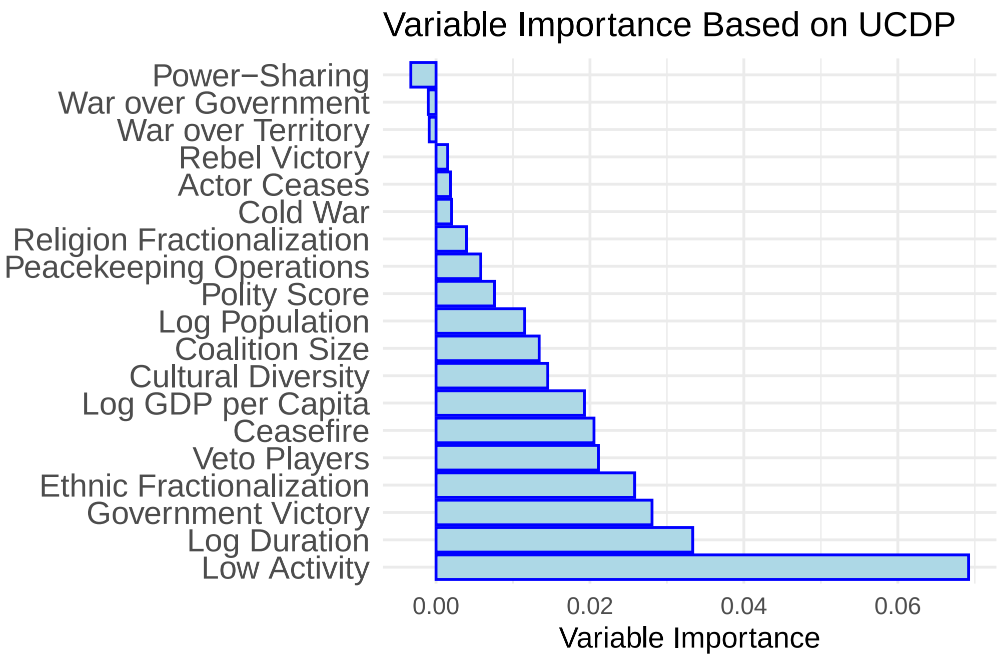
RSF Models

Var Imp with RSF: Any

```
# Applying labels
episodes_a <- apply_labels(episodes,
                           time_to_recur_any = "Time to Recur Any",
                           recur_any = "War Recurrence Any",
                           dis = "Actor Ceases",
                           cease = "Ceasefire",
                           govvic = "Government Victory",
                           rebvic = "Rebel Victory",
                           lowac = "Low Activity",
                           W4 = "Coalition Size",
                           pko_u = "Peacekeeping Operations",
                           log_dur = "Log Duration",
                           cold_war = "Cold War",
                           p_polity2 = "Polity Score",
                           fe_etfra = "Ethnic Fractionalization",
                           log_gdp = "Log GDP per Capita",
                           log_pop = "Log Population",
```

```r
                            ps_original = "Power-Sharing",
                            al_religion2000 = "Religion Fractionalization",
                            fe_cultdiv = "Cultural Diversity",
                            veto_u = "Veto Players",
                            gov_war = "War over Government",
                            ter_war = "War over Territory")

# Subsetting data
X <- subset(episodes_a, select = c(time_to_recur_any, recur_any, dis, cease, govvic, rebvic,
X <- na.omit(X)

# Ensure numeric variables are correctly formatted
X <- X %>%
  mutate(across(everything(), as.numeric))

# Define cross-validation folds
set.seed(123) # For reproducibility
folds <- createFolds(X$recur_any, k = 5, list = TRUE)

# Initialize a list to store variable importance scores
importance_list <- vector("list", length(folds))

# Perform cross-validation
for (i in seq_along(folds)) {
  # Define training and validation sets
  train_indices <- unlist(folds[-i])
  test_indices <- unlist(folds[i])

  train_data <- X[train_indices, ]
  test_data <- X[test_indices, ]

  # Train the Random Survival Forest model
  rsf <- rfsrc(Surv(time_to_recur_any, recur_any) ~ ., data = train_data, ntree = 1000, nodes

  # Store the variable importance scores
  importance_list[[i]] <- rsf$importance
}

# Aggregate variable importance scores
importance_df <- as.data.frame(do.call(cbind, importance_list))
importance_df$Variable <- rownames(importance_df)
rownames(importance_df) <- NULL
```

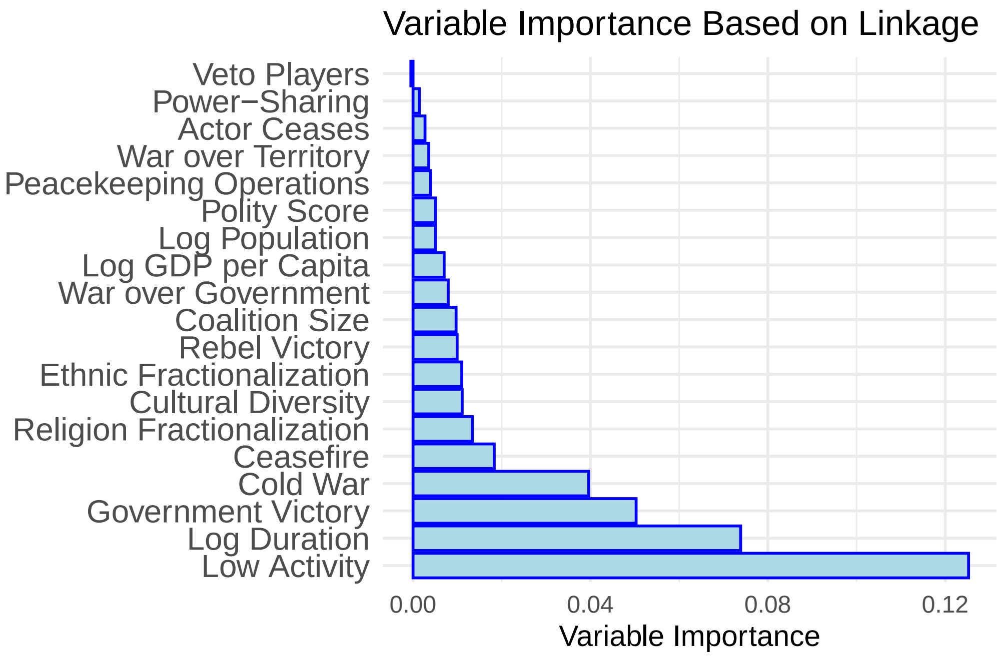
```r
importance_df$Importance <- rowMeans(importance_df[, -ncol(importance_df)])
importance_df <- importance_df[, c("Variable", "Importance")]

# Relabeling variables for plot
var_labels <- c("Actor Ceases", "Ceasefire", "Government Victory", "Rebel Victory", "Low Acti
names(var_labels) <- c("dis", "cease", "govvic", "rebvic", "lowac", "pko_u", "log_dur", "colo

# Adding variable labels
importance_df$Variable <- factor(importance_df$Variable, levels = names(var_labels), labels =

# Order by importance
importance_df <- importance_df %>% arrange(desc(Importance))
importance_df$Variable <- factor(importance_df$Variable, levels = importance_df$Variable)

# Plotting using ggplot2
ggplot(importance_df, aes(x = Importance, y = Variable)) +
  geom_bar(stat = "identity", fill = "lightblue", color = "blue") +
  theme_minimal() +
  labs(title = "Variable Importance Based on UCDP", x = "Variable Importance", y = "") +
  theme(axis.text.y = element_text(size = 12))
```



Var Imp with RSF: Link

```
# Applying labels
episodes_a <- apply_labels(episodes,
                           time_to_recur = "Time to Recur",
                           recur_side = "War Recurrence",
                           dis = "Actor Ceases",
                           cease = "Ceasefire",
                           govvic = "Government Victory",
                           rebvic = "Rebel Victory",
                           lowac = "Low Activity",
                           W4 = "Coalition Size",
                           pko_u = "Peacekeeping Operations",
                           log_dur = "Log Duration",
                           cold_war = "Cold War",
                           p_polity2 = "Polity Score",
                           fe_etfra = "Ethnic Fractionalization",
                           log_gdp = "Log GDP per Capita",
                           log_pop = "Log Population",
                           ps_original = "Power-Sharing",
                           al_religion2000 = "Religion Fractionalization",
                           fe_cultdiv = "Cultural Diversity",
                           veto_u = "Veto Players",
                           gov_war = "War over Government",
                           ter_war = "War over Territory")

# Subsetting data
X <- subset(episodes_a, select = c(time_to_recur, recur_side, dis, cease, govvic, rebvic, lo
X <- na.omit(X)

# Ensure numeric variables are correctly formatted
X <- X %>%
  mutate(across(everything(), as.numeric))

# Define cross-validation folds
set.seed(123) # For reproducibility
folds <- createFolds(X$recur_side, k = 5, list = TRUE)

# Initialize a list to store variable importance scores
importance_list <- vector("list", length(folds))

# Perform cross-validation
```

```
for (i in seq_along(folds)) {
  # Define training and validation sets
  train_indices <- unlist(folds[-i])
  test_indices <- unlist(folds[i])

  train_data <- X[train_indices, ]
  test_data <- X[test_indices, ]

  # Train the Random Survival Forest model
  rsf <- rfsrc(Surv(time_to_recur, recur_side) ~ ., data = train_data, ntree = 1000, nodesize

  # Store the variable importance scores
  importance_list[[i]] <- rsf$importance
}

# Aggregate variable importance scores
importance_df <- as.data.frame(do.call(cbind, importance_list))
importance_df$Variable <- rownames(importance_df)
rownames(importance_df) <- NULL
importance_df$Importance <- rowMeans(importance_df[, -ncol(importance_df)])
importance_df <- importance_df[, c("Variable", "Importance")]

# Relabeling variables for plot
var_labels <- c("Actor Ceases", "Ceasefire", "Government Victory", "Rebel Victory", "Low Acti
names(var_labels) <- c("dis", "cease", "govvic", "rebvic", "lowac", "pko_u", "log_dur", "colo

# Adding variable labels
importance_df$Variable <- factor(importance_df$Variable, levels = names(var_labels), labels =

# Order by importance
importance_df <- importance_df %>% arrange(desc(Importance))
importance_df$Variable <- factor(importance_df$Variable, levels = importance_df$Variable)

# Plotting using ggplot2
ggplot(importance_df, aes(x = Importance, y = Variable)) +
  geom_bar(stat = "identity", fill = "lightblue", color = "blue") +
  theme_minimal() +
  labs(title = "Variable Importance Based on Linkage", x = "Variable Importance", y = "") +
  theme(axis.text.y = element_text(size = 12))
```

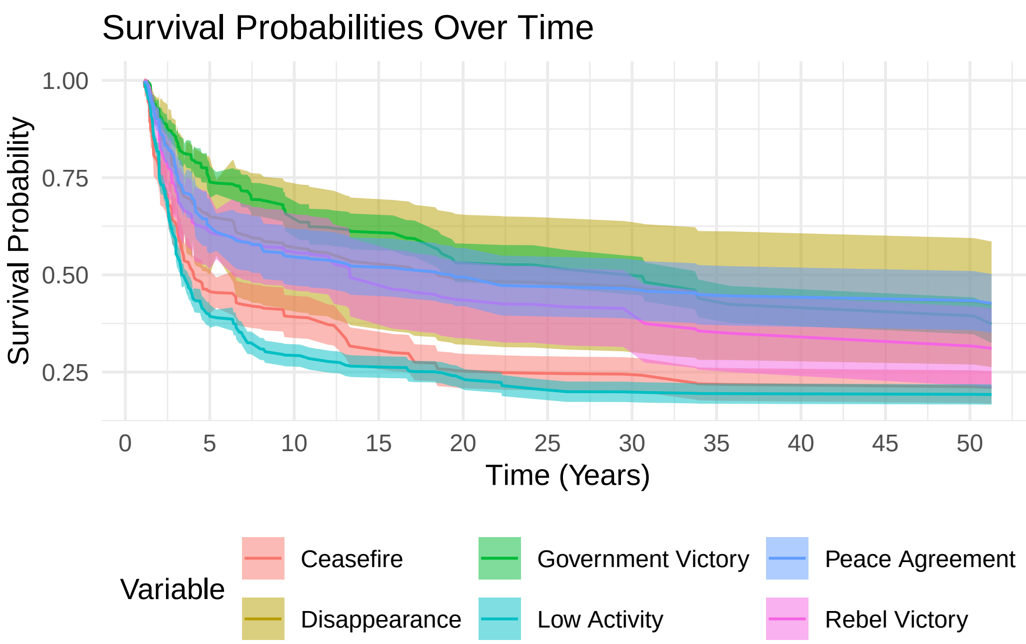


RSF- Any Survival with No CV

```
# Subsetting and cleaning data
X <- subset(episodes, select = c(time_to_recur_any, recur_any, dis, cease, govvic, rebvic, lo
                                  cold_war, p_polity2, fe_etfra, log_gdp, log_pop, ps_origin
                                  al_religion2000, fe_cultdiv, veto_u, gov_war, ter_war))
X <- na.omit(X)
X <- X %>% mutate(across(everything(), as.numeric))

# Convert time to years
X$time_to_recur_any <- X$time_to_recur_any / 365.25

# Train the Random Survival Forest model on the entire dataset
fit <- rfsrc(Surv(time_to_recur_any, recur_any) ~ ., data = X, ntree = 1000, nodesize = 5, ns

# Predict survival probabilities for the entire dataset
pred <- predict(fit, X, OOB = TRUE, type = "response")
survival_probs <- as.data.frame(pred$survival)

# Extract time points (already in years)
time_points <- fit$time.interest
```

```
colnames(survival_probs) <- paste("Time", round(time_points, 2), sep = "_")
survival_probs$id <- 1:nrow(survival_probs)
X$id <- 1:nrow(X)

# Convert survival probabilities to long format for plotting
survival_probs_long <- survival_probs %>%
  pivot_longer(cols = -id, names_to = "Time", values_to = "Probability") %>%
  mutate(Time = as.numeric(gsub("Time_", "", Time)))

# Combine with the original data to get the binary outcome variables
survival_probs_long <- left_join(survival_probs_long, X, by = "id")

# Function to calculate mean and confidence intervals
calculate_summary <- function(data) {
  data %>%
    group_by(Time) %>%
    summarise(
      Mean_Probability = mean(Probability, na.rm = TRUE),
      SD = sd(Probability, na.rm = TRUE),
      Lower_CI = Mean_Probability - qt(0.975, length(Probability)-1) * SD / sqrt(length(Proba
      Upper_CI = Mean_Probability + qt(0.975, length(Probability)-1) * SD / sqrt(length(Proba
    )
}

# Calculate summary statistics for each binary outcome variable
dis_summary <- calculate_summary(filter(survival_probs_long, dis == 1))
cease_summary <- calculate_summary(filter(survival_probs_long, cease == 1))
govvic_summary <- calculate_summary(filter(survival_probs_long, govvic == 1))
rebvic_summary <- calculate_summary(filter(survival_probs_long, rebvic == 1))
lowac_summary <- calculate_summary(filter(survival_probs_long, lowac == 1))
peace_summary <- calculate_summary(filter(survival_probs_long, peace == 1))

# Plot using ggplot2, one line per variable of interest with confidence intervals
ggplot() +
  geom_line(data = dis_summary, aes(x = Time, y = Mean_Probability, color = "Disappearance")
  geom_ribbon(data = dis_summary, aes(x = Time, ymin = Lower_CI, ymax = Upper_CI, fill = "Di
  geom_line(data = cease_summary, aes(x = Time, y = Mean_Probability, color = "Ceasefire")) 
  geom_ribbon(data = cease_summary, aes(x = Time, ymin = Lower_CI, ymax = Upper_CI, fill = "
  geom_line(data = govvic_summary, aes(x = Time, y = Mean_Probability, color = "Government V
  geom_ribbon(data = govvic_summary, aes(x = Time, ymin = Lower_CI, ymax = Upper_CI, fill = 
  geom_line(data = rebvic_summary, aes(x = Time, y = Mean_Probability, color = "Rebel Victor
  geom_ribbon(data = rebvic_summary, aes(x = Time, ymin = Lower_CI, ymax = Upper_CI, fill = 
```

```
  geom_line(data = lowac_summary, aes(x = Time, y = Mean_Probability, color = "Low Activity")
  geom_ribbon(data = lowac_summary, aes(x = Time, ymin = Lower_CI, ymax = Upper_CI, fill = "L
  geom_line(data = peace_summary, aes(x = Time, y = Mean_Probability, color = "Peace Agreemen
  geom_ribbon(data = peace_summary, aes(x = Time, ymin = Lower_CI, ymax = Upper_CI, fill = "P
  scale_x_continuous(breaks = seq(0, max(survival_probs_long$Time), by = 5)) +  # Set breaks
  labs(title = "Survival Probabilities Over Time", x = "Time (Years)", y = "Survival Probabil
  theme_minimal() +
  theme(legend.position = "bottom")
```

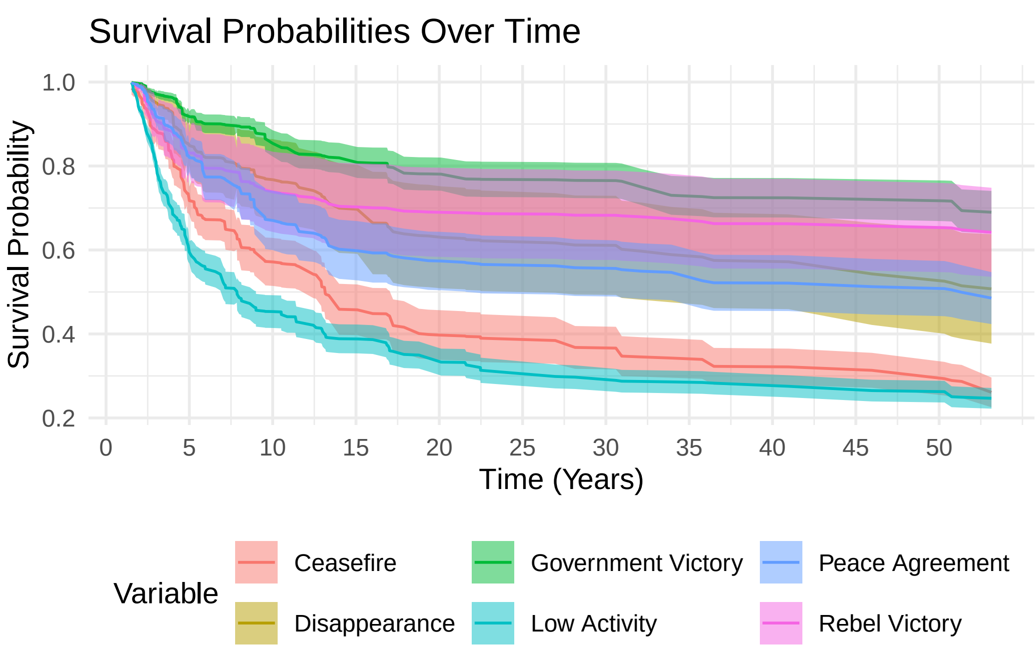


RSF- Side Survival with No CV

```
# Subsetting and cleaning data
X <- subset(episodes, select = c(time_to_recur, recur_side, dis, cease, govvic, rebvic, lowac
                                  cold_war, p_polity2, fe_etfra, log_gdp, log_pop, ps_origin
                                  al_religion2000, fe_cultdiv, veto_u, gov_war, ter_war))
X <- na.omit(X)
X <- X %>% mutate(across(everything(), as.numeric))

# Convert time to years
X$time_to_recur <- X$time_to_recur / 365.25
```

```
# Train the Random Survival Forest model on the entire dataset
fit <- rfsrc(Surv(time_to_recur, recur_side) ~ ., data = X, ntree = 1000, nodesize = 5, nspli

# Predict survival probabilities for the entire dataset
pred <- predict(fit, X, OOB = TRUE, type = "response")
survival_probs <- as.data.frame(pred$survival)

# Extract time points (already in years)
time_points <- fit$time.interest
colnames(survival_probs) <- paste("Time", round(time_points, 2), sep = "_")
survival_probs$id <- 1:nrow(survival_probs)
X$id <- 1:nrow(X)

# Convert survival probabilities to long format for plotting
survival_probs_long <- survival_probs %>%
  pivot_longer(cols = -id, names_to = "Time", values_to = "Probability") %>%
  mutate(Time = as.numeric(gsub("Time_", "", Time)))

# Combine with the original data to get the binary outcome variables
survival_probs_long <- left_join(survival_probs_long, X, by = "id")

# Function to calculate mean and confidence intervals
calculate_summary <- function(data) {
  data %>%
    group_by(Time) %>%
    summarise(
      Mean_Probability = mean(Probability, na.rm = TRUE),
      SD = sd(Probability, na.rm = TRUE),
      Lower_CI = Mean_Probability - qt(0.975, length(Probability)-1) * SD / sqrt(length(Proba
      Upper_CI = Mean_Probability + qt(0.975, length(Probability)-1) * SD / sqrt(length(Proba
    )
}

# Calculate summary statistics for each binary outcome variable
dis_summary <- calculate_summary(filter(survival_probs_long, dis == 1))
cease_summary <- calculate_summary(filter(survival_probs_long, cease == 1))
govvic_summary <- calculate_summary(filter(survival_probs_long, govvic == 1))
rebvic_summary <- calculate_summary(filter(survival_probs_long, rebvic == 1))
lowac_summary <- calculate_summary(filter(survival_probs_long, lowac == 1))
peace_summary <- calculate_summary(filter(survival_probs_long, peace == 1))

# Plot using ggplot2, one line per variable of interest with confidence intervals
```

```r
ggplot() +
  geom_line(data = dis_summary, aes(x = Time, y = Mean_Probability, color = "Disappearance"))
  geom_ribbon(data = dis_summary, aes(x = Time, ymin = Lower_CI, ymax = Upper_CI, fill = "Dis
  geom_line(data = cease_summary, aes(x = Time, y = Mean_Probability, color = "Ceasefire")) +
  geom_ribbon(data = cease_summary, aes(x = Time, ymin = Lower_CI, ymax = Upper_CI, fill = "C
  geom_line(data = govvic_summary, aes(x = Time, y = Mean_Probability, color = "Government Vi
  geom_ribbon(data = govvic_summary, aes(x = Time, ymin = Lower_CI, ymax = Upper_CI, fill = "
  geom_line(data = rebvic_summary, aes(x = Time, y = Mean_Probability, color = "Rebel Victory
  geom_ribbon(data = rebvic_summary, aes(x = Time, ymin = Lower_CI, ymax = Upper_CI, fill = "
  geom_line(data = lowac_summary, aes(x = Time, y = Mean_Probability, color = "Low Activity")
  geom_ribbon(data = lowac_summary, aes(x = Time, ymin = Lower_CI, ymax = Upper_CI, fill = "L
  geom_line(data = peace_summary, aes(x = Time, y = Mean_Probability, color = "Peace Agreemen
  geom_ribbon(data = peace_summary, aes(x = Time, ymin = Lower_CI, ymax = Upper_CI, fill = "P
  scale_x_continuous(breaks = seq(0, max(survival_probs_long$Time), by = 5)) +  # Set breaks
  labs(title = "Survival Probabilities Over Time", x = "Time (Years)", y = "Survival Probabil
  theme_minimal() +
  theme(legend.position = "bottom")
```



No Low Ac

```r
### remove low activity cases
episodes <- episodes %>% filter(lowac != 1)
```

Kaplan-Mier Curves

```r
# Create a new dataframe from episodes
new_episodes <- episodes

# Convert the numerical outcome variable to a factor with appropriate labels, excluding "Low
new_episodes$outcome <- factor(new_episodes$outcome, levels = c(1, 2, 3, 4, 6),
                               labels = c("Peace Agreement", "Ceasefire", "Gov Victory", "Rel

# Check the levels of the outcome variable
print(levels(new_episodes$outcome))
```

```
[1] "Peace Agreement" "Ceasefire"       "Gov Victory"     "Rebel Victory"
[5] "Actor Ceases"
```

```r
# Inspect the distribution of survival times
summary(new_episodes$time_to_recur_any)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    369    1365    4705    6941   10768   27774
```

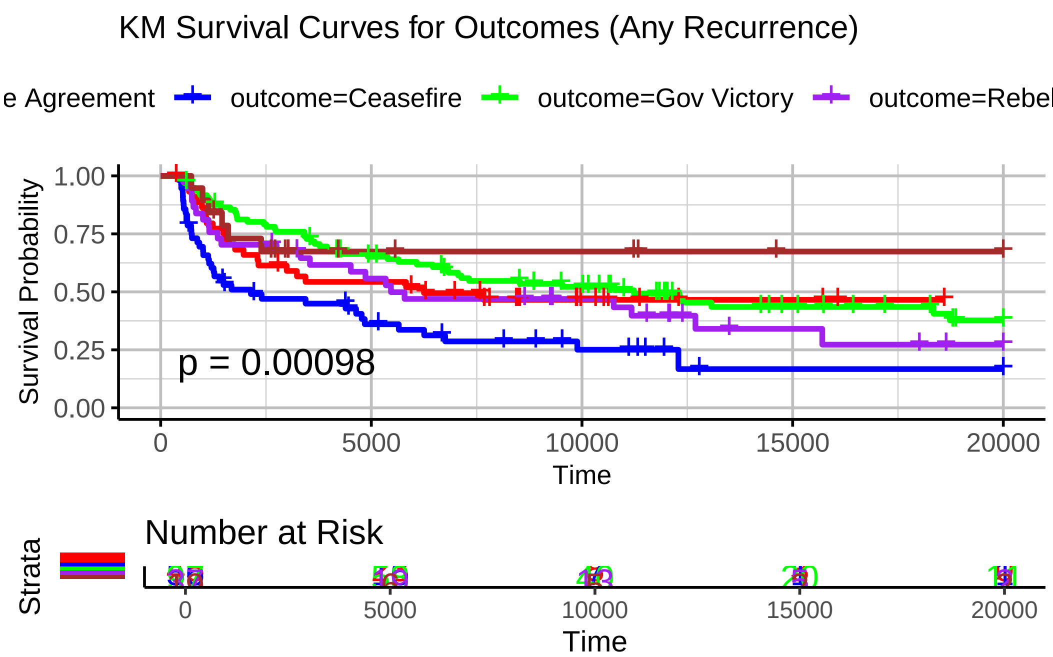
```r
summary(new_episodes$time_to_recur)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    560    3189    9357   10573   16567   28034
```

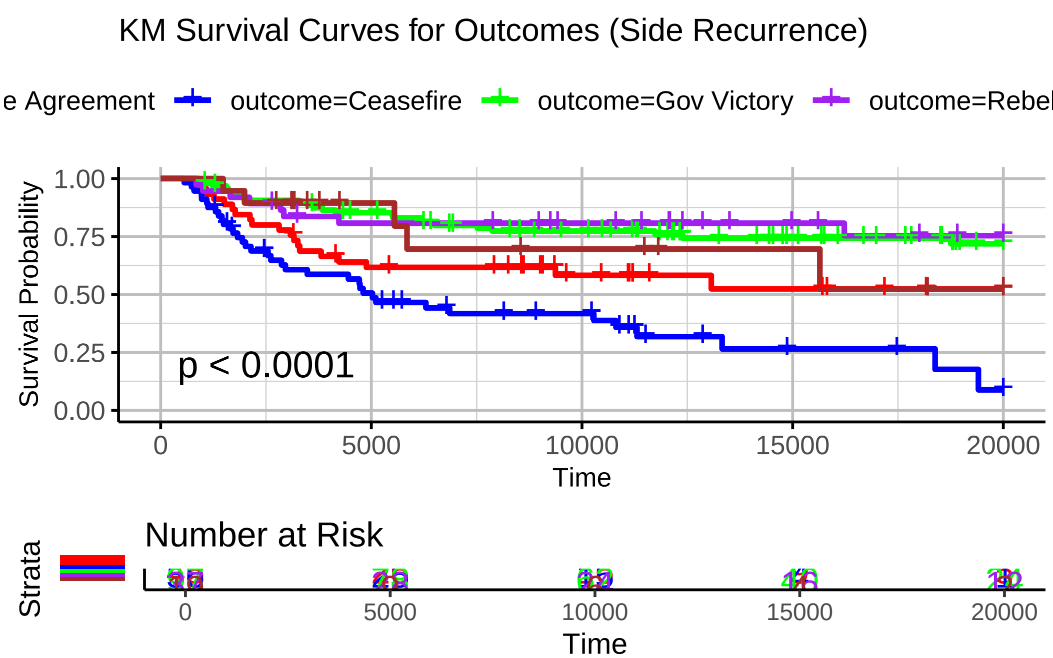
```r
# Trim follow-up time to 20,000 days
new_episodes$time_to_recur_any <- pmin(new_episodes$time_to_recur_any, 20000)
new_episodes$time_to_recur <- pmin(new_episodes$time_to_recur, 20000)

# Fit the Kaplan-Meier survival curves for the first plot
km_fit_outcome_any <- survfit(Surv(time_to_recur_any, recur_any) ~ outcome, data = new_episo
```

```
# Plot the Kaplan-Meier survival curves with percentages in the risk table
km_plot_any <- ggsurvplot(km_fit_outcome_any, data = new_episodes,
                          pval = TRUE, conf.int = FALSE,
                          risk.table = TRUE, risk.table.col = "strata",
                          risk.table.y.text.col = TRUE,
                          risk.table.height = 0.25,
                          risk.table.title = "Number at Risk",
                          ggtheme = theme_classic(),
                          palette = c("red", "blue", "green", "purple", "brown"),
                          title = "KM Survival Curves for Outcomes (Any Recurrence)",
                          xlab = "Time", ylab = "Survival Probability",
                          tables.y.text = FALSE)

# Customize the survival plot to add grid lines
km_plot_any$plot <- km_plot_any$plot +
    theme(legend.text = element_text(size = 10),  # Increase legend text size
          legend.title = element_text(size = 12),  # Increase legend title size
          axis.title.x = element_text(size = 10),  # X-axis title size
          axis.title.y = element_text(size = 10),  # Y-axis title size
          axis.text.x = element_text(size = 10),   # X-axis text size
          axis.text.y = element_text(size = 10),   # Y-axis text size
          plot.title = element_text(size = 12),    # Plot title size
          panel.grid.major = element_line(color = "gray", size = 0.5),  # Add major grid line
          panel.grid.minor = element_line(color = "lightgray", size = 0.25))  # Add minor gri

# Remove grid lines from the risk table
km_plot_any$risk.table <- km_plot_any$risk.table +
    theme(panel.grid.major = element_blank(),
          panel.grid.minor = element_blank())

# Fit the Kaplan-Meier survival curves for the second plot
km_fit_outcome_side <- survfit(Surv(time_to_recur, recur_side) ~ outcome, data = new_episodes

# Plot the Kaplan-Meier survival curves with percentages in the risk table for the second plo
km_plot_side <- ggsurvplot(km_fit_outcome_side, data = new_episodes,
                           pval = TRUE, conf.int = FALSE,
                           risk.table = TRUE, risk.table.col = "strata",
                           risk.table.y.text.col = TRUE,
                           risk.table.height = 0.25,
                           risk.table.title = "Number at Risk",
                           ggtheme = theme_classic(),
                           palette = c("red", "blue", "green", "purple", "brown"),
```

```
                            title = "KM Survival Curves for Outcomes (Side Recurrence)",
                            xlab = "Time", ylab = "Survival Probability",
                            tables.y.text = FALSE)

# Customize the survival plot to add grid lines
km_plot_side$plot <- km_plot_side$plot +
    theme(legend.text = element_text(size = 10),  # Increase legend text size
          legend.title = element_text(size = 12),  # Increase legend title size
          axis.title.x = element_text(size = 10),  # X-axis title size
          axis.title.y = element_text(size = 10),  # Y-axis title size
          axis.text.x = element_text(size = 10),   # X-axis text size
          axis.text.y = element_text(size = 10),   # Y-axis text size
          plot.title = element_text(size = 12),    # Plot title size
          panel.grid.major = element_line(color = "gray", size = 0.5),  # Add major grid line
          panel.grid.minor = element_line(color = "lightgray", size = 0.25))  # Add minor gri

# Remove grid lines from the risk table
km_plot_side$risk.table <- km_plot_side$risk.table +
    theme(panel.grid.major = element_blank(),
          panel.grid.minor = element_blank())

# Print the plots
print(km_plot_any)
```




```
print(km_plot_side)
```



Cox models

```
## Models

m1 <- coxph(Surv(time_to_recur_any, recur_any) ~ dis + cease + govvic + rebvic, data = episod

m2 <- coxph(Surv(time_to_recur_any, recur_any) ~ dis + cease + govvic + rebvic + pko_u + log_

m3 <- coxph(Surv(time_to_recur, recur_side) ~ dis + cease + govvic + rebvic, data = episodes

m4 <- coxph(Surv(time_to_recur, recur_side) ~ dis + cease + govvic + rebvic + pko_u + log_du

stargazer(m1, m2, m3, m4, type = "text")
```

```
==========================================================================================
                                          Dependent variable:
                     ---------------------------------------------------------------------
                               time_to_recur_any                      time_to_recur
                            (1)                   (2)                 (3)                 (4)
------------------------------------------------------------------------------------------
dis                       -0.472                0.349              -0.440              -0.160
                         (0.459)               (0.667)            (0.503)             (0.747)

cease                     0.620**               1.351**            0.678**              0.778
                         (0.263)               (0.445)            (0.284)             (0.471)

govvic                    -0.222                0.370              -0.790**            -0.252
                         (0.253)               (0.455)            (0.309)             (0.528)

rebvic                     0.086                0.831              -0.924**            -0.227
                         (0.296)               (0.521)            (0.423)             (0.696)

pko_u                                          -0.552                                  -0.691
                                               (0.462)                                (0.527)

log_dur                                         0.082*                                  0.065
                                               (0.045)                                (0.061)

cold_war                                       -0.420                                  -1.074**
                                               (0.275)                                (0.358)
```

log_gdp		0.003		-0.110
		(0.090)		(0.113)
veto_u		0.390**		0.344*
		(0.146)		(0.166)
gov_war		0.401		-0.264
		(0.266)		(0.302)
ter_war				
		(0.000)		(0.000)
p_polity2		0.010		-0.045
		(0.037)		(0.046)
fe_etfra		0.496		-0.451
		(1.001)		(1.168)
W4		-0.861		1.523
		(1.122)		(1.487)
log_pop		0.117*		-0.032
		(0.071)		(0.088)
al_language2000		1.737**		2.493***
		(0.700)		(0.891)
ps_original		0.055		-0.137
		(0.349)		(0.418)
al_religion2000		-0.690		-0.916
		(0.544)		(0.663)
fe_cultdiv		-1.729		-1.586
		(0.995)		(1.263)

Observations	254	173	254	173
R2	0.062	0.251	0.134	0.291
Max. Possible R2	0.996	0.995	0.977	0.977
Log Likelihood	-705.245	-440.117	-458.157	-297.331
Wald Test	15.440*** (df = 4)	64.930*** (df = 18)	42.750*** (df = 4)	64.860*** (df

```
LR Test              16.391*** (df = 4) 49.916*** (df = 18) 36.449*** (df = 4) 59.464*** (df
Score (Logrank) Test 18.506*** (df = 4) 54.963*** (df = 18) 43.412*** (df = 4) 61.667*** (df
==============================================================================================
Note:                                                                 *p<0.1; **p<0.05; ***p
```

```r
stargazer(m1, m2, m3, m4,
          ord.intercepts = TRUE,
          dep.var.labels = c("UCDP ID-Based War Recurrence", "UCDP ID-Based War Recurrence",
          covariate.labels = c("Actor Ceases", "Ceasefire", "Government Victory", "Rebel Vic
          type = "text")
```

```
==============================================================================================
                                                        Dependent variable:
                           -------------------------------------------------------------------
                                UCDP ID-Based War Recurrence              UCDP ID-Based War Recu
                                  (1)                 (2)                 (3)
----------------------------------------------------------------------------------------------
Actor Ceases                     -0.472              0.349               -0.440              -(
                                (0.459)             (0.667)             (0.503)              ((

Ceasefire                        0.620**             1.351**             0.678**              (
                                (0.263)             (0.445)             (0.284)              ((

Government Victory               -0.222              0.370               -0.790**            -(
                                (0.253)             (0.455)             (0.309)              ((

Rebel Victory                     0.086              0.831               -0.924**            -(
                                (0.296)             (0.521)             (0.423)              ((

Peacekeeping Missions                                -0.552                                  -(
                                                    (0.462)                                  ((

Log(Duration)                                        0.082*                                   (
                                                    (0.045)                                  ((

Cold War                                             -0.420                                  -1
                                                    (0.275)                                  ((

Log(GDP per Capita)                                  0.003                                   -(
                                                    (0.090)                                  ((
```

Number of Veto Players		0.390**		0
		(0.146)		(0
War over Government		0.401		-0
		(0.266)		(0
War over Territory				
		(0.000)		(0
Polity Score		0.010		-0
		(0.037)		(0
Ethnic Fractionalization		0.496		-0
		(1.001)		(1
Coalition Size		-0.861		1
		(1.122)		(1
Log(Population)		0.117*		-0
		(0.071)		(0
Language Fractionalization		1.737**		2.4
		(0.700)		(0
Power-Sharing		0.055		-0
		(0.349)		(0
Religion Fractionalization		-0.690		-0
		(0.544)		(0
Cultural Diversity		-1.729		-1
		(0.995)		(1
Observations	254	173	254	
R2	0.062	0.251	0.134	0
Max. Possible R2	0.996	0.995	0.977	0
Log Likelihood	-705.245	-440.117	-458.157	-29
Wald Test	15.440*** (df = 4)	64.930*** (df = 18)	42.750*** (df = 4)	64.860**
LR Test	16.391*** (df = 4)	49.916*** (df = 18)	36.449*** (df = 4)	59.464**
Score (Logrank) Test	18.506*** (df = 4)	54.963*** (df = 18)	43.412*** (df = 4)	61.667**

Note: *p<0.1; **p<0.05

Coefficient Plots fom Cox

```
# Extracting model coefficients and robust standard errors from the summary
coef_summary <- summary(m2)$coefficients

# Rename terms for better readability
terms_rename <- c(
  "dis" = "Actor Ceases",
  "cease" = "Ceasefire",
  "govvic" = "Government Victory",
  "rebvic" = "Rebel Victory",
  "pko_u" = "Peacekeeping Missions",
  "log_dur" = "Log(Duration)",
  "cold_war" = "Cold War",
  "log_gdp" = "Log(GDP per Capita)",
  "veto_u" = "Number of Veto Players",
  "gov_war" = "War over Government",
  "ter_war" = "War over Territory",
  "p_polity2" = "Polity Score",
  "fe_etfra" = "Ethnic Fractionalization",
  "W4" = "Coalition Size",
  "log_pop" = "Log(Population)",
  "al_language2000" = "Language Fractionalization",
  "ps_original" = "Power-Sharing",
  "al_religion2000" = "Religion Fractionalization",
  "fe_cultdiv" = "Cultural Diversity"
)

# Create a dataframe for plotting
df_coef <- data.frame(
  Term = rownames(coef_summary),
  Estimate = coef_summary[, "coef"],
  StdErr = coef_summary[, "robust se"],
  Lower = coef_summary[, "coef"] - 1.96 * coef_summary[, "robust se"],
  Upper = coef_summary[, "coef"] + 1.96 * coef_summary[, "robust se"],
  p_value = coef_summary[, "Pr(>|z|)"]
)

# Apply the renaming to the Term column
df_coef$Term <- terms_rename[df_coef$Term]

# Plotting using ggplot2
```

```
library(ggplot2)
cofp_any<-ggplot(df_coef, aes(x = Estimate, y = Term)) +
  geom_point() +
  geom_errorbarh(aes(xmin = Lower, xmax = Upper), height = 0.2, color = ifelse(df_coef$p_valu
  theme_minimal() +
  labs(title = "Coefficient Plot of Cox Model: UCDP",
       x = "Coefficient Value",
       y = "Variables")

# Extracting model coefficients and robust standard errors from the summary
coef_summary <- summary(m4)$coefficients

# Create a dataframe for plotting
df_coef <- data.frame(
  Term = rownames(coef_summary),
  Estimate = coef_summary[, "coef"],
  StdErr = coef_summary[, "robust se"],
  Lower = coef_summary[, "coef"] - 1.96 * coef_summary[, "robust se"],
  Upper = coef_summary[, "coef"] + 1.96 * coef_summary[, "robust se"],
  p_value = coef_summary[, "Pr(>|z|)"]
)

# Apply the renaming to the Term column
df_coef$Term <- terms_rename[df_coef$Term]

# Plotting using ggplot2
cofp_side<-ggplot(df_coef, aes(x = Estimate, y = Term)) +
  geom_point() +
  geom_errorbarh(aes(xmin = Lower, xmax = Upper), height = 0.2, color = ifelse(df_coef$p_valu
  theme_minimal() +
  labs(title = "Coefficient Plot of Cox Model: Link",
       x = "Coefficient Value",
       y = "Variables")
```

Survival rates from Cox

```
# Define factor levels and labels without "Low Activity"
outcome_levels <- c("Peace Agreement", "Ceasefire", "Gov Victory", "Rebel Victory", "Actor Ce

# Convert outcome variable to factor with specified levels in the original dataset
```

```r
episodes$outcomes <- factor(episodes$outcome, levels = c(1, 2, 3, 4, 6), labels = outcome_lev

m2 <- coxph(Surv(time_to_recur_any, recur_any) ~ as.factor(outcomes) + pko_u + log_dur + colo

m4 <- coxph(Surv(time_to_recur, recur_side) ~ as.factor(outcomes) + pko_u + log_dur + cold_wa

representative_data <- episodes %>%
  summarise(
    pko_u = median(as.numeric(pko_u), na.rm = TRUE),
    log_dur = mean(log_dur, na.rm = TRUE),
    cold_war = median(cold_war, na.rm = TRUE),
    log_gdp = mean(log_gdp, na.rm = TRUE),
    veto_u = median(veto_u, na.rm = TRUE),
    gov_war = median(gov_war, na.rm = TRUE),
    ter_war = median(ter_war, na.rm = TRUE),
    p_polity2 = mean(p_polity2, na.rm = TRUE),
    fe_etfra = mean(fe_etfra, na.rm = TRUE),
    W4 = mean(W4, na.rm = TRUE),
    log_pop = mean(log_pop, na.rm = TRUE),
    al_language2000 = mean(al_language2000, na.rm = TRUE),
    ps_original = median(ps_original, na.rm = TRUE),
    al_religion2000 = mean(al_religion2000, na.rm = TRUE),
    fe_cultdiv = mean(fe_cultdiv, na.rm = TRUE)
  )

# Expand the data to include different outcomes
expanded_data <- do.call(rbind, replicate(5, representative_data, simplify = FALSE))
expanded_data$outcomes <- factor(rep(outcome_levels, each = 1), levels = outcome_levels)

# Generate survival curves based on the Cox model for m2
surv_fits_any <- survfit(m2, newdata = expanded_data)

# Define colors for the plot
colors <- c("Peace Agreement" = "red", "Ceasefire" = "blue", "Gov Victory" = "green", "Rebel

# Plotting survival curves for Any Recurrence
plot_any <- ggsurvplot(
  surv_fits_any,
  data = expanded_data,
  pval = TRUE,  # Show p-value for log-rank test
  conf.int = FALSE,  # Show confidence intervals
  xlab = "Time in Days",
```

```r
  ylab = "Survival Probability",
  ggtheme = theme_bw(),
  xlim = c(0, 20000),
  break.time.by = 1000,
  risk.table = TRUE,  # Show number at risk table
  surv.scale = "percent",
  legend.title = "Outcome",
  legend.labs = levels(expanded_data$outcomes),  # Automatically use factor labels
  title = "Survival Probability Over Time for War Recurrence Based on ACD Conflict ID",
  subtitle = "",
  caption = "",
  palette = colors,
  size = 1.2,  # Increase line width for better visibility
  linetype = "solid"  # Set line type to solid for all curves
)
```

```
Warning in .pvalue(fit, data = data, method = method, pval = pval, pval.coord = pval.coord,
 This is a null model.
```

```
Warning: No shared levels found between `names(values)` of the manual scale and the
data's fill values.
```

```
Warning in (function (survsummary, times, survtable = c("cumevents",
"risk.table", : The length of legend.labs should be 1
```

```r
# Customize the legend font size and axis titles for Any Recurrence plot
plot_any$plot <- plot_any$plot +
  theme(legend.text = element_text(size = 12),  # Increase legend text size
        legend.title = element_text(size = 14),  # Increase legend title size
        axis.title.x = element_text(size = 10),  # X-axis title size
        axis.title.y = element_text(size = 10),  # Y-axis title size
        axis.text.x = element_text(size = 10, angle = 45, hjust = 1),  # X-axis text size anc
        axis.text.y = element_text(size = 10),   # Y-axis text size
        plot.title = element_text(size = 12))    # Plot title size

# Customize the risk table text for Any Recurrence plot
plot_any$table <- plot_any$table +
  theme(axis.text.x = element_text(size = 10, angle = 45, hjust = 1))  # Rotate risk table te

# Generate survival curves based on the Cox model for m4
surv_fits_side <- survfit(m4, newdata = expanded_data)
```

```
# Plotting survival curves for Side Recurrence
plot_side <- ggsurvplot(
  surv_fits_side,
  data = expanded_data,
  pval = TRUE,  # Show p-value for log-rank test
  conf.int = FALSE,  # Show confidence intervals
  xlab = "Time in Days",
  ylab = "Survival Probability",
  ggtheme = theme_bw(),
  xlim = c(0, 20000),
  break.time.by = 1000,
  risk.table = TRUE,  # Show number at risk table
  surv.scale = "percent",
  legend.title = "Outcome",
  legend.labs = levels(expanded_data$outcomes),  # Automatically use factor labels
  title = "Survival Probability Over Time for War Recurrence Based on Sufficient Linkage",
  subtitle = "",
  caption = "",
  palette = colors,
  size = 1.2,  # Increase line width for better visibility
  linetype = "solid"  # Set line type to solid for all curves
)
```

```
Warning in .pvalue(fit, data = data, method = method, pval = pval, pval.coord = pval.coord,
 This is a null model.
```

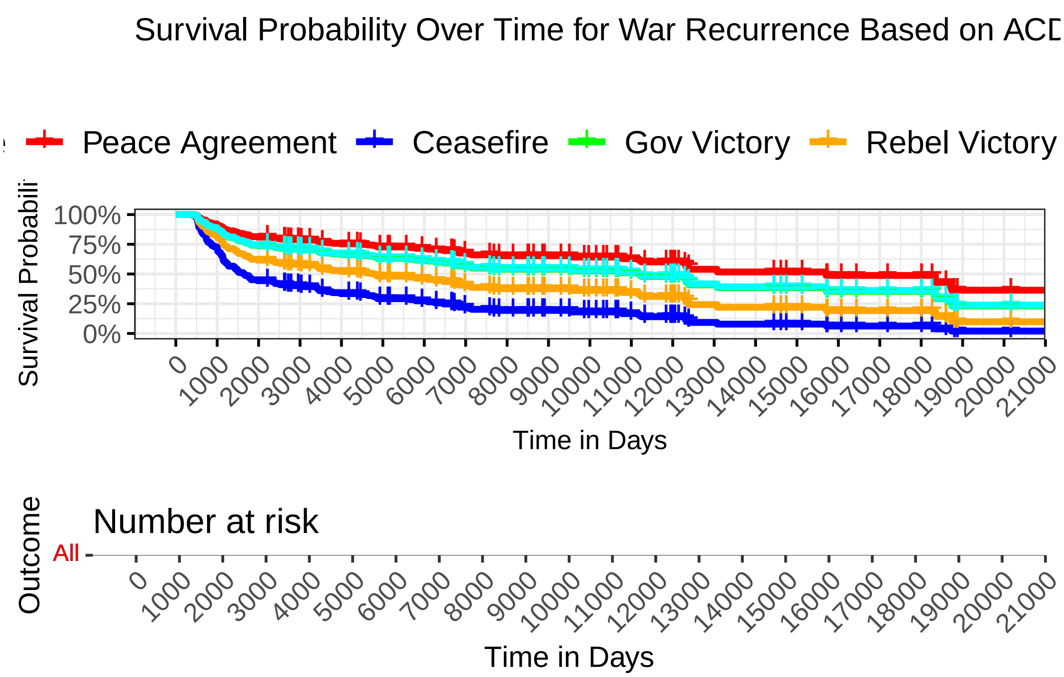
```
Warning: No shared levels found between `names(values)` of the manual scale and the
data's fill values.
```

```
Warning in (function (survsummary, times, survtable = c("cumevents",
"risk.table", : The length of legend.labs should be 1
```

```
# Customize the legend font size and axis titles for Side Recurrence plot
plot_side$plot <- plot_side$plot +
  theme(legend.text = element_text(size = 12),  # Increase legend text size
        legend.title = element_text(size = 14),  # Increase legend title size
        axis.title.x = element_text(size = 10),  # X-axis title size
        axis.title.y = element_text(size = 10),  # Y-axis title size
        axis.text.x = element_text(size = 10, angle = 45, hjust = 1),  # X-axis text size and
        axis.text.y = element_text(size = 10),   # Y-axis text size
```

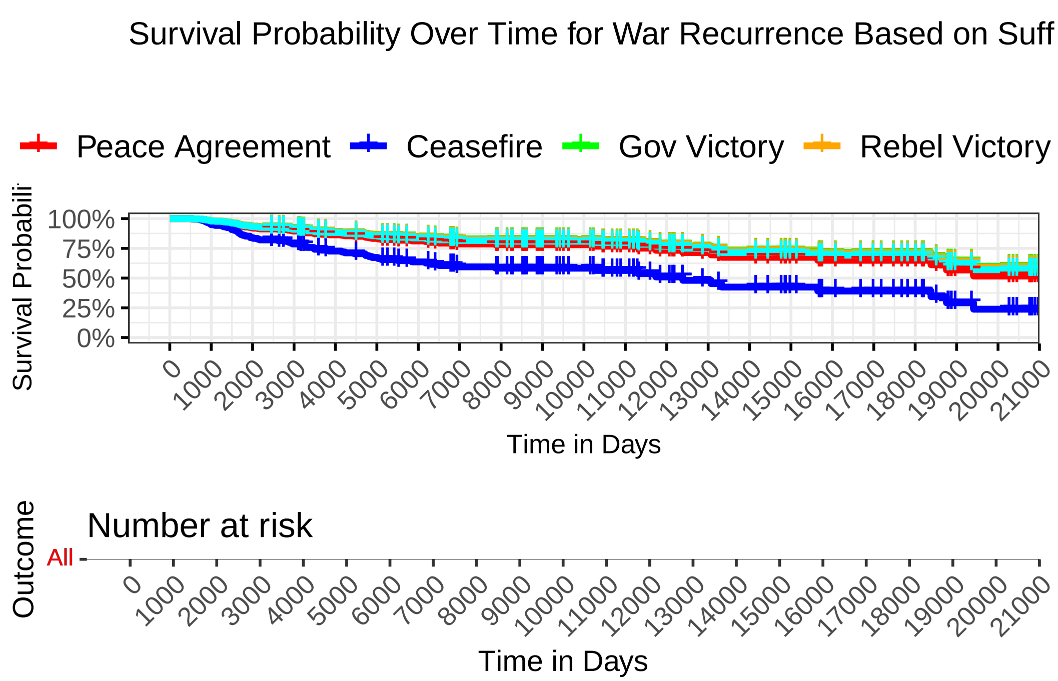
```
       plot.title = element_text(size = 12))     # Plot title size

# Customize the risk table text for Side Recurrence plot
plot_side$table <- plot_side$table +
  theme(axis.text.x = element_text(size = 10, angle = 45, hjust = 1))  # Rotate risk table te

# Print the plots
print(plot_any)
```

```
Warning: No shared levels found between `names(values)` of the manual scale and the
data's fill values.

Warning: No shared levels found between `names(values)` of the manual scale and the
data's fill values.
No shared levels found between `names(values)` of the manual scale and the
data's fill values.

Warning: No shared levels found between `names(values)` of the manual scale and the
data's colour values.

Warning: No shared levels found between `names(values)` of the manual scale and the
data's fill values.
```



```r
print(plot_side)
```

```
Warning: No shared levels found between `names(values)` of the manual scale and the
data's fill values.
No shared levels found between `names(values)` of the manual scale and the
data's fill values.
No shared levels found between `names(values)` of the manual scale and the
data's fill values.

Warning: No shared levels found between `names(values)` of the manual scale and the
data's colour values.

Warning: No shared levels found between `names(values)` of the manual scale and the
data's fill values.
```

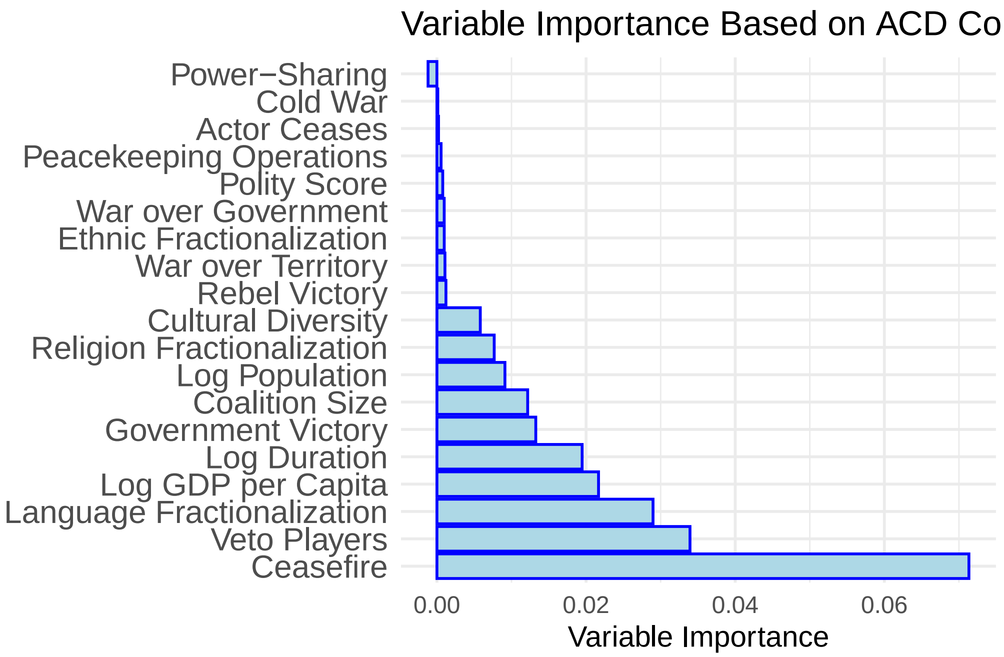


RSF- Var Im-Any

```r
# Install and load necessary packages
# install.packages("randomForestSRC")
# install.packages("timeROC")
# install.packages("caret")
library(randomForestSRC)
library(readxl)
library(pec)
library(timeROC)
library(survivalROC)
library(ggplot2)
library(expss)
library(survival)
library(dplyr)
library(caret)

# Assuming 'episodes' is already loaded

# Applying labels
episodes_a <- apply_labels(episodes,
                           time_to_recur_any = "Time to Recur Any",
                           recur_any = "War Recurrence Any",
                           dis = "Actor Ceases",
                           cease = "Ceasefire",
                           govvic = "Government Victory",
                           rebvic = "Rebel Victory",
                           lowac = "Low Activity",
                           W4 = "Coalition Size",
                           pko_u = "Peacekeeping Operations",
                           log_dur = "Log Duration",
                           cold_war = "Cold War",
                           p_polity2 = "Polity Score",
                           fe_etfra = "Ethnic Fractionalization",
                           log_gdp = "Log GDP per Capita",
                           log_pop = "Log Population",
                           al_language2000 = "Language Fractionalization",
                           ps_original = "Power-Sharing",
                           al_religion2000 = "Religion Fractionalization",
                           fe_cultdiv = "Cultural Diversity",
                           veto_u = "Veto Players",
                           gov_war = "War over Government",
                           ter_war = "War over Territory")
```

```
# Subsetting data and removing lowac
X <- subset(episodes_a, select = c(time_to_recur_any, recur_any, dis, cease, govvic, rebvic, 
X <- na.omit(X)

# Ensure numeric variables are correctly formatted
X <- X %>%
  mutate(across(everything(), as.numeric))

# Define cross-validation folds
set.seed(123) # For reproducibility
folds <- createFolds(X$recur_any, k = 5, list = TRUE)

# Initialize a list to store variable importance scores
importance_list <- vector("list", length(folds))

# Perform cross-validation
for (i in seq_along(folds)) {
  # Define training and validation sets
  train_indices <- unlist(folds[-i])
  test_indices <- unlist(folds[i])

  train_data <- X[train_indices, ]
  test_data <- X[test_indices, ]

  # Train the Random Survival Forest model
  rsf <- rfsrc(Surv(time_to_recur_any, recur_any) ~ ., data = train_data, ntree = 1000, nodes

  # Store the variable importance scores
  importance_list[[i]] <- rsf$importance
}

# Aggregate variable importance scores
importance_df <- as.data.frame(do.call(cbind, importance_list))
importance_df$Variable <- rownames(importance_df)
rownames(importance_df) <- NULL
importance_df$Importance <- rowMeans(importance_df[, -ncol(importance_df)])
importance_df <- importance_df[, c("Variable", "Importance")]

# Relabeling variables for plot
var_labels <- c("Actor Ceases", "Ceasefire", "Government Victory", "Rebel Victory", "Peaceke
names(var_labels) <- c("dis", "cease", "govvic", "rebvic", "pko_u", "log_dur", "cold_war", "l
```

```
# Adding variable labels
importance_df$Variable <- factor(importance_df$Variable, levels = names(var_labels), labels =

# Order by importance
importance_df <- importance_df %>% arrange(desc(Importance))
importance_df$Variable <- factor(importance_df$Variable, levels = importance_df$Variable)

# Plotting using ggplot2
ggplot(importance_df, aes(x = Importance, y = Variable)) +
  geom_bar(stat = "identity", fill = "lightblue", color = "blue") +
  theme_minimal() +
  labs(title = "Variable Importance Based on ACD Conflict ID", x = "Variable Importance", y =
  theme(axis.text.y = element_text(size = 12))
```



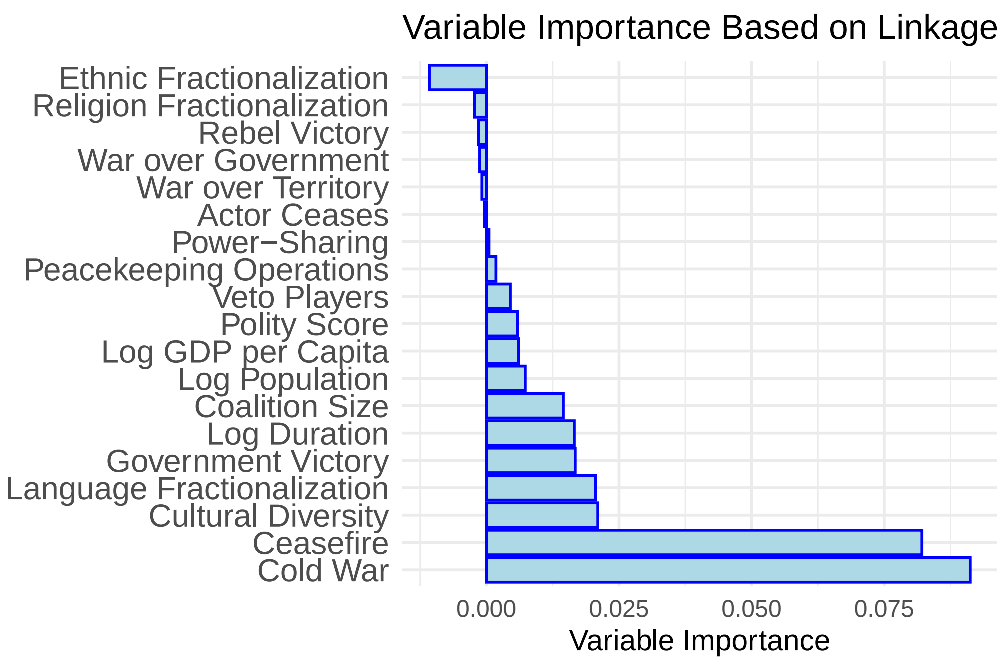
RSF - Var Im - Link

```
# Install and load necessary packages
# install.packages("randomForestSRC")
# install.packages("timeROC")
# install.packages("caret")
```

```r
library(randomForestSRC)
library(readxl)
library(pec)
library(timeROC)
library(survivalROC)
library(ggplot2)
library(expss)
library(survival)
library(dplyr)
library(caret)

# Assuming 'episodes' is already loaded

# Applying labels
episodes_a <- apply_labels(episodes,
                           time_to_recur = "Time to Recur",
                           recur_any = "War Recurrence",
                           dis = "Actor Ceases",
                           cease = "Ceasefire",
                           govvic = "Government Victory",
                           rebvic = "Rebel Victory",
                           W4 = "Coalition Size",
                           pko_u = "Peacekeeping Operations",
                           log_dur = "Log Duration",
                           cold_war = "Cold War",
                           p_polity2 = "Polity Score",
                           fe_etfra = "Ethnic Fractionalization",
                           log_gdp = "Log GDP per Capita",
                           log_pop = "Log Population",
                           al_language2000 = "Language Fractionalization",
                           ps_original = "Power-Sharing",
                           al_religion2000 = "Religion Fractionalization",
                           fe_cultdiv = "Cultural Diversity",
                           veto_u = "Veto Players",
                           gov_war = "War over Government",
                           ter_war = "War over Territory")

# Subsetting data
X <- subset(episodes_a, select = c(time_to_recur, recur_any, dis, cease, govvic, rebvic, pko_
X <- na.omit(X)

# Ensure numeric variables are correctly formatted
```

```r
X <- X %>%
  mutate(across(everything(), as.numeric))

# Define cross-validation folds
set.seed(123) # For reproducibility
folds <- createFolds(X$recur_any, k = 5, list = TRUE)

# Initialize a list to store variable importance scores
importance_list <- vector("list", length(folds))

# Perform cross-validation
for (i in seq_along(folds)) {
  # Define training and validation sets
  train_indices <- unlist(folds[-i])
  test_indices <- unlist(folds[i])

  train_data <- X[train_indices, ]
  test_data <- X[test_indices, ]

  # Train the Random Survival Forest model
  rsf <- rfsrc(Surv(time_to_recur, recur_any) ~ ., data = train_data, ntree = 1000, nodesize

  # Store the variable importance scores
  importance_list[[i]] <- rsf$importance
}

# Aggregate variable importance scores
importance_df <- as.data.frame(do.call(cbind, importance_list))
importance_df$Variable <- rownames(importance_df)
rownames(importance_df) <- NULL
importance_df$Importance <- rowMeans(importance_df[, -ncol(importance_df)])
importance_df <- importance_df[, c("Variable", "Importance")]

# Relabeling variables for plot
var_labels <- c("Actor Ceases", "Ceasefire", "Government Victory", "Rebel Victory", "Peacekee
names(var_labels) <- c("dis", "cease", "govvic", "rebvic", "pko_u", "log_dur", "cold_war", "l

# Adding variable labels
importance_df$Variable <- factor(importance_df$Variable, levels = names(var_labels), labels =

# Order by importance
importance_df <- importance_df %>% arrange(desc(Importance))
```

```r
importance_df$Variable <- factor(importance_df$Variable, levels = importance_df$Variable)

# Plotting using ggplot2
ggplot(importance_df, aes(x = Importance, y = Variable)) +
  geom_bar(stat = "identity", fill = "lightblue", color = "blue") +
  theme_minimal() +
  labs(title = "Variable Importance Based on Linkage", x = "Variable Importance", y = "") +
  theme(axis.text.y = element_text(size = 12))
```

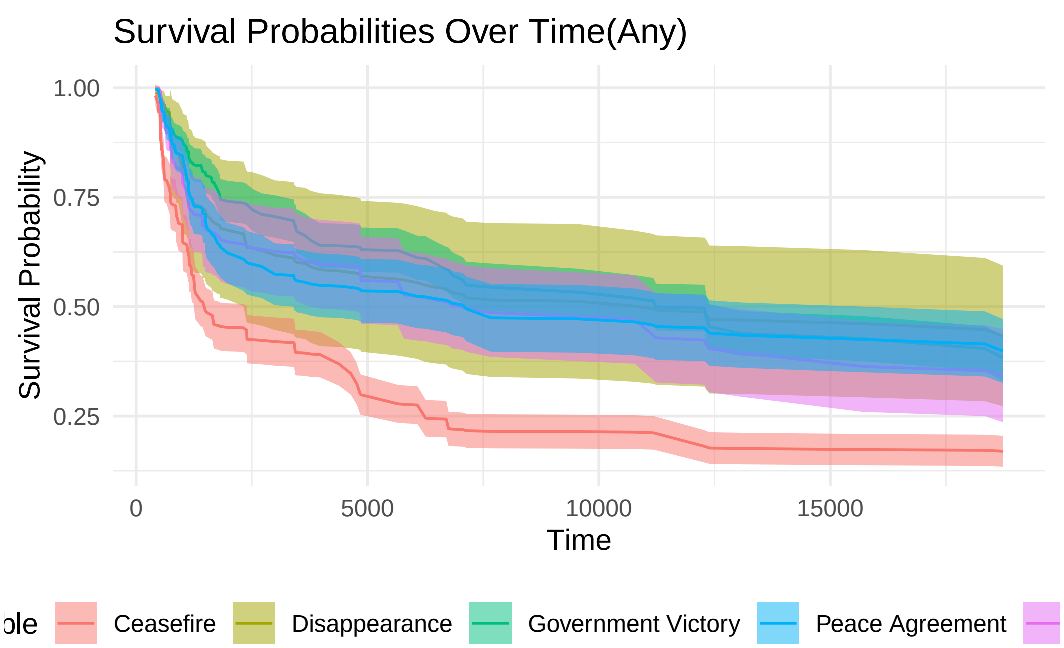


RSF -Sur Probs - Any

```r
# Install and load necessary packages
# install.packages("randomForestSRC")
# install.packages("ggplot2")
library(randomForestSRC)
library(ggplot2)
library(dplyr)
library(tidyr)

# Assuming 'episodes' is your dataset
```

```r
# Subsetting and cleaning data
X <- subset(episodes, select = c(time_to_recur_any, recur_any, dis, cease, govvic, rebvic, pe
                                   cold_war, p_polity2, fe_etfra, log_gdp, log_pop, al_langua
                                   al_religion2000, fe_cultdiv, veto_u, gov_war, ter_war))
X <- na.omit(X)
X <- X %>% mutate(across(everything(), as.numeric))

# Train the Random Survival Forest model on the entire dataset
fit <- rfsrc(Surv(time_to_recur_any, recur_any) ~ ., data = X, ntree = 1000, nodesize = 5, ns

# Predict survival probabilities for the entire dataset
pred <- predict(fit, X, OOB = TRUE, type = "response")
survival_probs <- as.data.frame(pred$survival)

# Extract time points
time_points <- fit$time.interest
colnames(survival_probs) <- paste("Time", time_points, sep = "_")
survival_probs$id <- 1:nrow(survival_probs)
X$id <- 1:nrow(X)

# Convert survival probabilities to long format for plotting
survival_probs_long <- survival_probs %>%
  pivot_longer(cols = -id, names_to = "Time", values_to = "Probability") %>%
  mutate(Time = as.numeric(gsub("Time_", "", Time)))

# Combine with the original data to get the binary outcome variables
survival_probs_long <- left_join(survival_probs_long, X, by = "id")

# Function to calculate mean and confidence intervals
calculate_summary <- function(data) {
  data %>%
    group_by(Time) %>%
    summarise(
      Mean_Probability = mean(Probability, na.rm = TRUE),
      SD = sd(Probability, na.rm = TRUE),
      Lower_CI = Mean_Probability - qt(0.975, length(Probability)-1) * SD / sqrt(length(Proba
      Upper_CI = Mean_Probability + qt(0.975, length(Probability)-1) * SD / sqrt(length(Proba
    )
}

# Calculate summary statistics for each binary outcome variable
dis_summary <- calculate_summary(filter(survival_probs_long, dis == 1))
```

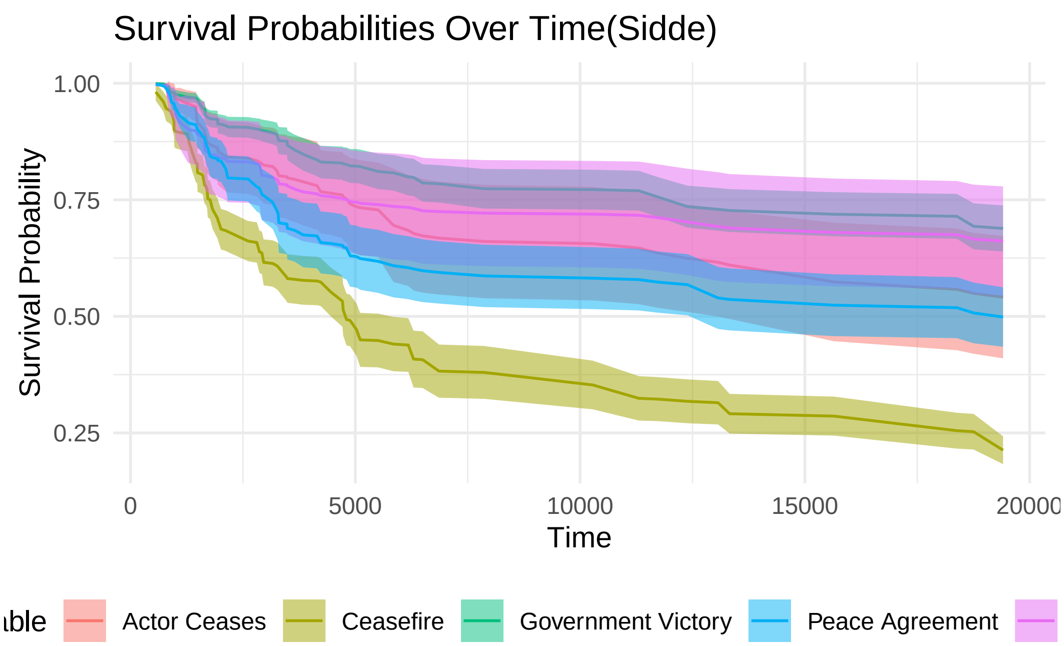
```
cease_summary <- calculate_summary(filter(survival_probs_long, cease == 1))
govvic_summary <- calculate_summary(filter(survival_probs_long, govvic == 1))
rebvic_summary <- calculate_summary(filter(survival_probs_long, rebvic == 1))
peace_summary <- calculate_summary(filter(survival_probs_long, peace == 1))

# Plot using ggplot2, one line per variable of interest with confidence intervals
ggplot() +
  geom_line(data = dis_summary, aes(x = Time, y = Mean_Probability, color = "Disappearance"))
  geom_ribbon(data = dis_summary, aes(x = Time, ymin = Lower_CI, ymax = Upper_CI, fill = "Dis
  geom_line(data = cease_summary, aes(x = Time, y = Mean_Probability, color = "Ceasefire")) +
  geom_ribbon(data = cease_summary, aes(x = Time, ymin = Lower_CI, ymax = Upper_CI, fill = "C
  geom_line(data = govvic_summary, aes(x = Time, y = Mean_Probability, color = "Government Vi
  geom_ribbon(data = govvic_summary, aes(x = Time, ymin = Lower_CI, ymax = Upper_CI, fill = "
  geom_line(data = rebvic_summary, aes(x = Time, y = Mean_Probability, color = "Rebel Victory
  geom_ribbon(data = rebvic_summary, aes(x = Time, ymin = Lower_CI, ymax = Upper_CI, fill = "
  geom_line(data = peace_summary, aes(x = Time, y = Mean_Probability, color = "Peace Agreemen
  geom_ribbon(data = peace_summary, aes(x = Time, ymin = Lower_CI, ymax = Upper_CI, fill = "P
  labs(title = "Survival Probabilities Over Time(Any)", x = "Time", y = "Survival Probability
  theme_minimal() +
  theme(legend.position = "bottom")
```



RSF - Sur Probs- Side

```
# Install and load necessary packages
# install.packages("randomForestSRC")
# install.packages("ggplot2")
library(randomForestSRC)
library(ggplot2)
library(dplyr)
library(tidyr)

# Assuming 'episodes' is your dataset

# Subsetting and cleaning data
X <- subset(episodes, select = c(time_to_recur, recur_side, dis, cease, govvic, rebvic, peace
                                   cold_war, p_polity2, fe_etfra, log_gdp, log_pop, al_langua
                                   al_religion2000, fe_cultdiv, veto_u, gov_war, ter_war))
X <- na.omit(X)
X <- X %>% mutate(across(everything(), as.numeric))

# Train the Random Survival Forest model on the entire dataset
fit <- rfsrc(Surv(time_to_recur, recur_side) ~ ., data = X, ntree = 1000, nodesize = 5, nspli

# Predict survival probabilities for the entire dataset
pred <- predict(fit, X, OOB = TRUE, type = "response")
survival_probs <- as.data.frame(pred$survival)

# Extract time points
time_points <- fit$time.interest
colnames(survival_probs) <- paste("Time", time_points, sep = "_")
survival_probs$id <- 1:nrow(survival_probs)
X$id <- 1:nrow(X)

# Convert survival probabilities to long format for plotting
survival_probs_long <- survival_probs %>%
  pivot_longer(cols = -id, names_to = "Time", values_to = "Probability") %>%
  mutate(Time = as.numeric(gsub("Time_", "", Time)))

# Combine with the original data to get the binary outcome variables
survival_probs_long <- left_join(survival_probs_long, X, by = "id")

# Function to calculate mean and confidence intervals
calculate_summary <- function(data) {
```

```
  data %>%
    group_by(Time) %>%
    summarise(
      Mean_Probability = mean(Probability, na.rm = TRUE),
      SD = sd(Probability, na.rm = TRUE),
      Lower_CI = Mean_Probability - qt(0.975, length(Probability)-1) * SD / sqrt(length(Proba
      Upper_CI = Mean_Probability + qt(0.975, length(Probability)-1) * SD / sqrt(length(Proba
    )
}

# Calculate summary statistics for each binary outcome variable
dis_summary <- calculate_summary(filter(survival_probs_long, dis == 1))
cease_summary <- calculate_summary(filter(survival_probs_long, cease == 1))
govvic_summary <- calculate_summary(filter(survival_probs_long, govvic == 1))
rebvic_summary <- calculate_summary(filter(survival_probs_long, rebvic == 1))
peace_summary <- calculate_summary(filter(survival_probs_long, peace == 1))

# Plot using ggplot2, one line per variable of interest with confidence intervals
ggplot() +
  geom_line(data = dis_summary, aes(x = Time, y = Mean_Probability, color = "Actor Ceases"))
  geom_ribbon(data = dis_summary, aes(x = Time, ymin = Lower_CI, ymax = Upper_CI, fill = "Act
  geom_line(data = cease_summary, aes(x = Time, y = Mean_Probability, color = "Ceasefire")) +
  geom_ribbon(data = cease_summary, aes(x = Time, ymin = Lower_CI, ymax = Upper_CI, fill = "C
  geom_line(data = govvic_summary, aes(x = Time, y = Mean_Probability, color = "Government Vi
  geom_ribbon(data = govvic_summary, aes(x = Time, ymin = Lower_CI, ymax = Upper_CI, fill = "
  geom_line(data = rebvic_summary, aes(x = Time, y = Mean_Probability, color = "Rebel Victory
  geom_ribbon(data = rebvic_summary, aes(x = Time, ymin = Lower_CI, ymax = Upper_CI, fill = "
  geom_line(data = peace_summary, aes(x = Time, y = Mean_Probability, color = "Peace Agreemen
  geom_ribbon(data = peace_summary, aes(x = Time, ymin = Lower_CI, ymax = Upper_CI, fill = "P
  labs(title = "Survival Probabilities Over Time(Sidde)", x = "Time", y = "Survival Probabili
  theme_minimal() +
  theme(legend.position = "bottom")
```



Causal Identification

Synthetic Control

```r
# Create a subset with only the relevant variables
episodes_control <- subset(episodes, select = c("episode_id", "year", "recur_any",
                                                "log_gdp", "log_pop", "p_polity2",
                                                "fe_etfra", "log_dur", "cease"))
```

```r
# Remove rows with missing values
episodes_control <- na.omit(episodes_control)
```

```r
# Check the structure of the dataset
str(episodes_control)
```

```
tibble [184 x 9] (S3: tbl_df/tbl/data.frame)
 $ episode_id: num [1:184] 4 17 18 27 31 34 40 48 51 59 ...
 $ year      : num [1:184] 1968 1996 1998 1990 2012 ...
 $ recur_any : num [1:184] 0 1 1 0 1 1 1 0 1 1 ...
 $ log_gdp   : num [1:184] 5.28 7.2 6.91 7.27 7.06 ...
```

```
 $ log_pop   : num [1:184] 15.3 18.1 18.1 15.2 17.7 ...
 $ p_polity2 : num [1:184] -4 8 8 2 -3 -4 -6 -5 8 -5 ...
 $ fe_etfra  : num [1:184] 0.743 0.161 0.161 0.132 0.522 ...
 $ log_dur   : num [1:184] 5.29 9.16 3.61 0 8.37 ...
 $ cease     : num [1:184] 0 1 1 0 1 0 0 0 0 1 ...
 - attr(*, "na.action")= 'omit' Named int [1:70] 1 2 3 5 6 7 8 9 12 13 ...
  ..- attr(*, "names")= chr [1:70] "1" "2" "3" "5" ...
```

```r
# Check for missing values again to ensure they've been removed
sum(is.na(episodes_control))
```

```
[1] 0
```

Matching