Supplement 4

HIV-SN incidence analysis

Peter Kamerman and Prinisha Pillay

18 May 2019

Contents

Import data

```
data <- read_rds('data-cleaned/clean_data.rds') %>%
    # Select columns
    select(ID, visit_number, visit_day,
           hivsn_present, pain)
```

Process data

Estimate SN onset date

We estimated the date of SN onset (*'approximate day'*) as the mid-point between two successive visits when a participant was found to have transitioned between SN:no to SN:yes. Otherwise, *'approximate day'* was taken to be the visit_day.

```
### Create filter of participants who developed SN
sn_filter <- unique(data$ID[data$hivsn_present == 'yes'])

# SN:yes group
approx_sn <- data %>%
    # Filter for participants that developed SN
    filter(ID %in% sn_filter) %>%
    # Group by ID
    group_by(ID) %>%
    # Approximate days
    mutate(approximate_day = ifelse(visit_number == 1,
                                    yes = 0,
                                    no = ifelse(hivsn_present == 'yes',
                                                yes = (visit_day + lag(visit_day)) / 2,
                                                no = visit_day)),
           approximate_day = as.integer(round(approximate_day))) %>%
    # Add person years
    mutate(visit_year = visit_day / 365.25,
           approximate_year = approximate_day / 365.25)

# SN:no group
approx_no_sn <- data %>%
    # Filter for participants that did not develop SN
    filter(!ID %in% sn_filter) %>%
    # Group by ID
    group_by(ID) %>%
    # approximate days (because no SN developed, use visit_day)
    mutate(approximate_day = visit_day,
           approximate_day = as.integer(round(approximate_day))) %>%
    # Add person years
    mutate(visit_year = visit_day / 365.25,
           approximate_year = approximate_day / 365.25)

# Put approx_* data frames together
data <- bind_rows(approx_sn, approx_no_sn)
```

Cumulative incidence data

Extract 3 and 6 month periods for cumulative incidence.
Note: Data were censored at last screening within each 3/6 month period.

```
## Generate time filters
months_6 <- round(dyears(0.5) / ddays(1))
months_3 <- round(dyears(0.25) / ddays(1))

# Extract SN:yes within 6-month period
```

```
data_6months <- data %>%
    # Filter values within 6 months
    filter(visit_day <= months_6) %>%
    # Group by ID
    group_by(ID) %>%
    # Filter max visit_day
    filter(visit_day == max(visit_day))

# Extract SN:yes within first 3-month period
data_3months <- data %>%
    # Filter values within 6 months
    filter(visit_day <= months_3) %>%
    # Group by ID
    group_by(ID) %>%
    # Filter max visit_day
    filter(visit_day == max(visit_day))

# Extract SN:yes within second 3-month period
data_3to6months <-  data_6months %>%
    # Filter out those with SN with first 3 months
    filter(!ID %in% data_3months[data_3months$hivsn_present == 'yes', ]$ID)
```

Incidence rate data

Only need full 6-month period

```
# Extract SN:yes for full period
df_6months <- data %>%
    # Group by ID
    group_by(ID) %>%
    # Filter max visit_day
    filter(visit_day == max(visit_day))
```

Survival analysis data

```
# Create a data frame of SN:yes patients
data_sn.yes <- data %>%
    # Select columns
    select(ID,
           visit_number,
           visit_day,
           visit_year,
           approximate_day,
           approximate_year,
           hivsn_present,
           pain) %>%
    # Add counting columns
    ## SN:yes and SN:no
    mutate(sn_count = ifelse(hivsn_present == 'yes',
                             yes = 1,
                             no = 0)) %>%
    # Create counting index to identify when SN first develop
    ## Sum sn_count
    group_by(ID) %>%
    mutate(sn_count = cumsum(sn_count)) %>%
    # Filter by sn_count == 1 to get when SN first developed
    filter(sn_count == 1) %>%
    # Remove sn_count
    select(-sn_count)

# Create data frame of SN:no patients
data_sn.no <- data %>%
    # Select columns
    select(ID,
           visit_number,
           visit_day,
           visit_year,
           approximate_day,
           approximate_year,
           hivsn_present,
           pain) %>%
    # Filter out SN:yes patient with the filter created earlier
    filter(!ID %in% sn_filter) %>%
    # Filter out 'repeats' (only the data at the last visit)
    group_by(ID) %>%
    mutate(max_visits = max(visit_number)) %>%
    filter(visit_number == max_visits) %>%
    select(-max_visits)

# Merge the SN:yes and SN:no data frames
data_surv <- data_sn.yes %>%
    full_join(data_sn.no) %>%
    # New column with recoded hivsn_present data for survival analysis
    mutate(hivsn_coded = ifelse(hivsn_present == 'yes',
                                yes = 1, # SN:yes
                                no = 0)) # SN:no

# Add a new column with painful SN (but only if they have SN)
data_surv <- data_surv %>%
    mutate(hivsn_painful = ifelse(hivsn_present == 'yes' & pain == 'yes',
                                  yes = 'yes',
                                  no = ifelse(hivsn_present == 'yes' &
                                                  pain == 'no',
                                              yes = 'no',
                                              no = NA)))
```

Modelling data

```
# Identify and extract information on SN development (at anytime)
# by looking at the presence of SN at the final visit
data_model <- data %>%
    select(ID, visit_number, hivsn_present) %>%
    group_by(ID) %>%
    mutate(max_visit = max(visit_number)) %>%
    filter(visit_number == max_visit) %>%
    select(ID, hivsn_present) %>%
    rename(sn = hivsn_present)

# Join data_sn to data
data_model <- read_rds('data-cleaned/clean_data.rds') %>%
    left_join(data_model) %>%
    select(-hivsn_present)

# Restrict data to the baseline visit (visit 1)
data_model %<>%
    filter(visit_number == 1)

# Clean-up df_model data frame for analyses
## These data will be used for all the baseline charateristic analyses.
data_model %<>%
    # Recode rifafour 'prophylaxis' to 'yes'
    mutate(rifafour_treatment = as.character(rifafour_treatment),
           rifafour_treatment = ifelse(rifafour_treatment == 'prophylaxis',
                                       yes = 'yes',
                                       no = rifafour_treatment),
           rifafour_treatment = factor(rifafour_treatment)) %>%
    # Select data that will be modelled
    select(sn, age_years, sex, mass_kg, height_m, CD4_cell.ul,
           viral_load_copies.ml, alcohol_units.week, TB_current,
           rifafour_treatment)
```

Analysis

Cumulative incidence

Cumulative incidence measures the number of new cases per person in the population over a defined period of time (i.e., a fixed follow-up period). Therefore, to calculate cumulative incidence we defined a fixed 6-month follow-up period, and also subdivided this period into a 1st and 2nd 3-month period of this follow-up. To standardize the periods of follow-up, data were cleaned such that only visits falling into the indicated periods ((0-3 months], (3-6 months], (0-6 months]) were used to define SN status. As such, participant's whose last clinic visit occurred after 91 days (end of first 3-month interval) or 182 days (end of 6-month interval), and who were found to have new-onset SN at this visit, were recorded as SN:no over the 3 or 6-month period of follow-up. This conservative strategy may have lead to a slight under-estimation of the cumulative incidence of SN.

Boostrap function

```
## Formula for cases / person year
cases_boot <- function(data, i){
    foo <- data[i, ]
    tab <- table(foo$hivsn_present)
    prop <- round(prop.table(tab) * 100)
    case <- prop * 10
    case[2]
}
```

Full six-month period

```
# Tabulate SN:yes/SN:no
tab_sn <- table(data_6months$hivsn_present)

# Calculate proportions, convert to percent
prop_sn <- round(prop.table(tab_sn) * 100)

# Bootstrap cases per 1000 patients
## Method: BCa, resamples: 1999
cases_ci <- boot.ci(boot.out = boot(data = data_6months,
                                    statistic = cases_boot,
                                    R = 1999,
                                    stype = 'i'),
                    type = 'bca')

# Create summary table
as.data.frame(tab_sn) %>%
    rename(Count = Freq) %>%
    left_join(as.data.frame(prop_sn)) %>%
    rename(Percentage = Freq,
           sn_present = Var1) %>%
    bind_cols(`Cases/1000 patients` = c('', paste(cases_ci$t0))) %>%
    bind_cols(`Conf.interval` = c('', paste(cases_ci$bca[4],' - ', cases_ci$bca[5])))
```

```
##   sn_present Count Percentage Cases/1000 patients Conf.interval
## 1         no   103         86
## 2        yes    17         14                 140    80  -  210
```

First 3-month period

```
# Tabulate SN:yes/SN:no
tab_sn <- table(data_3months$hivsn_present)

# Calculate proportions, convert to percent
prop_sn <- round(prop.table(tab_sn) * 100)

# Bootstrap cases per 1000 patients
## Method: BCa, resamples: 1999
cases_ci <- boot.ci(boot.out = boot(data = data_3months,
                                    statistic = cases_boot,
                                    R = 1999,
                                    stype = 'i'),
                    type = 'bca')

# Create summary table
as.data.frame(tab_sn) %>%
    rename(Count = Freq) %>%
    left_join(as.data.frame(prop_sn)) %>%
    rename(Percentage = Freq,
           sn_present = Var1) %>%
    bind_cols(`Cases/1000 patients` = c('', paste(cases_ci$t0))) %>%
    bind_cols(`Conf.interval` = c('', paste0(cases_ci$bca[4],' to ', cases_ci$bca[5])))
```

```
##   sn_present Count Percentage Cases/1000 patients Conf.interval
## 1         no   113         94                                  
## 2        yes     7          6                  60     20 to 100
```

Second 3-month period

```
# Tabulate SN:yes/SN:no
tab_sn <- table(data_3to6months$hivsn_present)

# Calculate proportions, convert to percent
prop_sn <- round(prop.table(tab_sn) * 100)

# Bootstrap cases per 1000 patients
## Method: BCa, resamples: 1999
cases_ci <- boot.ci(boot.out = boot(data = data_3to6months,
                                    statistic = cases_boot,
                                    R = 1999,
                                    stype = 'i'),
                    type = 'bca')

# Create summary table
as.data.frame(tab_sn) %>%
    rename(Count = Freq) %>%
    left_join(as.data.frame(prop_sn)) %>%
    rename(Percentage = Freq,
           sn_present = Var1) %>%
    bind_cols(`Cases/1000 patients` = c('', paste(cases_ci$t0))) %>%
    bind_cols(`Conf.interval` = c('', paste0(cases_ci$bca[4],' to ', cases_ci$bca[5])))
```

```
##   sn_present Count Percentage Cases/1000 patients Conf.interval
## 1         no   103         91                                  
## 2        yes    10          9                  90     40 to 140
```

Incidence rate (person years)

Incidence rate is a measure of the number of new cases per unit of time. We did not have extact dates of SN onset to define the per patient unit of time, and nor was there uniform spacing between clinic visits. We therefore chose to calculate an approximate SN onset time, arbitrarily defined as the number of days between the first neuropathy screening and the mid-point between the visit when neuropathy was detected and the preceeding visit. For participants who did not develop SN, the date of censoring was defined by the number of days between the first neuropathy screening and the last screening (study exit).

Boostrap function

```
## Formula for cases / person year
boot_df <- function(data, i){
    foo <- data[i, ]
    cases <- sum(foo$hivsn_present == 'yes')
    person_time <- sum(foo$approximate_year)
    cases / person_time
}
```

Six-month period

```
# Bootstrap confidence intervals
df_ci <- boot.ci(boot.out = boot(data = df_6months,
                                 statistic = boot_df,
                                 R = 1999,
                                 stype = 'i'),
                 type = 'bca')

tibble(`Cases/person.year` = round(df_ci$t0, 2),
       Conf.interval = paste0(round(df_ci$bca[4], 2), ' to ',
                              round(df_ci$bca[5], 2)))
```

```
## # A tibble: 1 x 2
##   `Cases/person.year` Conf.interval
##                 <dbl> <chr>
## 1                0.37 0.24 to 0.55
```

Survival curves

Plot a basic Kaplan-Meyer survival curve

```
# Basic Kaplan-Meyer (KM)
## No predictors (~ 1)
km_basic <- survfit(Surv(time = approximate_day,
                         event = hivsn_coded) ~ 1,
                    data = data_surv)

# Summary of KM fit
km_basic
```

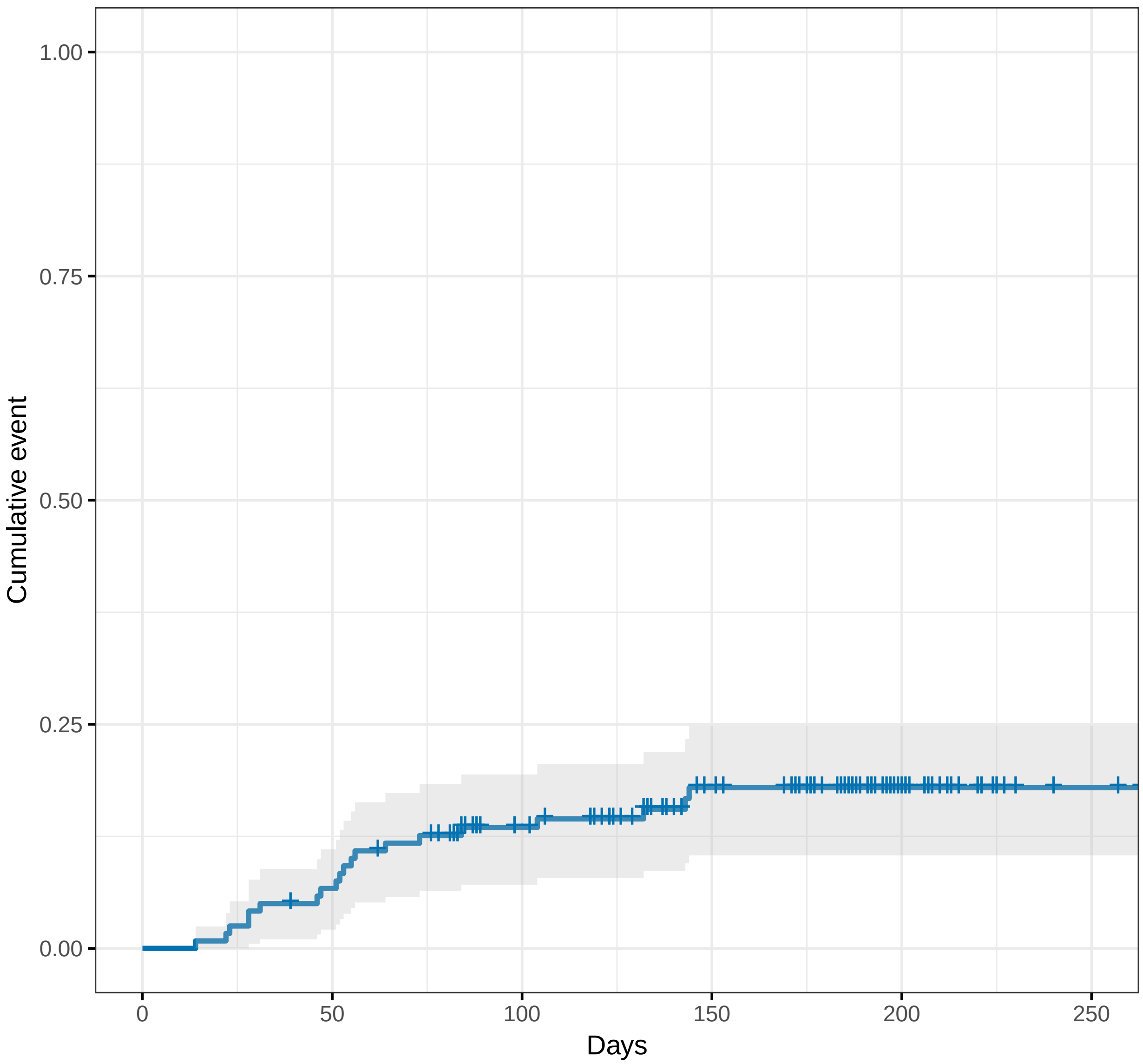
```
## Call: survfit(formula = Surv(time = approximate_day, event = hivsn_coded) ~
##     1, data = data_surv)
##
##       n  events  median 0.95LCL 0.95UCL
##     120      20      NA      NA      NA
```

```
summary(km_basic)
```

```
## Call: survfit(formula = Surv(time = approximate_day, event = hivsn_coded) ~
##     1, data = data_surv)
##
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##    14    120       1    0.992  0.0083        0.976        1.000
##    22    119       1    0.983  0.0117        0.961        1.000
##    23    118       1    0.975  0.0143        0.947        1.000
##    28    117       2    0.958  0.0182        0.923        0.995
##    31    115       1    0.950  0.0199        0.912        0.990
##    46    113       1    0.942  0.0214        0.901        0.985
##    47    112       1    0.933  0.0228        0.890        0.979
##    51    111       1    0.925  0.0241        0.879        0.973
##    52    110       1    0.916  0.0253        0.868        0.967
##    53    109       1    0.908  0.0264        0.858        0.961
##    55    108       1    0.900  0.0275        0.847        0.955
##    56    107       1    0.891  0.0285        0.837        0.949
##    64    105       1    0.883  0.0295        0.827        0.942
##    73    104       1    0.874  0.0304        0.817        0.936
##    84     98       1    0.865  0.0314        0.806        0.929
##   104     89       1    0.856  0.0325        0.794        0.922
##   132     80       1    0.845  0.0338        0.781        0.914
##   143     70       1    0.833  0.0354        0.766        0.905
##   144     69       1    0.821  0.0369        0.752        0.896
```

```
# Plot
ggsurvplot(km_basic,
           conf.int = TRUE,
           fun = 'event',
           risk.table = 'abs_pct',
           cumcensor = TRUE,
           cumevents = TRUE,
           ggtheme = theme_bw(),
           tables.theme = theme(plot.title = element_text(size = 12,
                                                          face = 'bold'),
                                panel.border = element_blank(),
                                panel.grid.major.x = element_blank(),
                                axis.title = element_blank(),
                                axis.text.x = element_blank(),
                                axis.ticks = element_blank()),
           fontsize = 4,
           tables.height = 0.1,
           legend = 'none',
           palette = '#0072B2',
           ylim = c(0, 1),
           xlab = 'Days',
           title = 'Kaplan-Meier Survival Curve: SN vs Days')
```

Kaplan−Meier Survival Curve: SN vs Days



Number at risk: n (%)

All	.20 (100)	111 (92)	91 (76)	66 (55)	25 (21)	2 (2)

Cumulative number of events

All	0	8	16	20	20	20

Cumulative number of censoring

All	0	1	13	34	76	98

```
# Publication plot
## New plot theme
theme_new <- function(){
  theme_bw(base_size = 18) +
  theme(axis.text = element_text(size = 22,
                                 colour = '#000000'),
        axis.title = element_text(size = 24,
                                  colour = '#000000'),
        panel.grid.major.x = element_line(size = 1,
                                          colour = '#CCCCCC'),
        panel.border = element_rect(size = 1.5),
        panel.grid.major.y = element_blank(),
        panel.grid.minor.y = element_blank(),
        panel.grid.minor.x = element_blank())
    }

## Plot
gg_plot <- ggsurvplot(km_basic,
                      fun = 'event',
                      conf.int = TRUE,
                      conf.int.fill = '#666666',
                      risk.table = 'abs_pct',
                      cumcensor = TRUE,
                      cumevents = TRUE,
                      tables.theme = theme(plot.title = element_text(size = 22,
                                                                     face = 'bold'),
                                           panel.border = element_blank(),
                                           panel.grid.major.x = element_blank(),
                                           axis.title = element_blank(),
                                           axis.text = element_blank(),
                                           axis.ticks = element_blank()),
                      fontsize = 7,
                      risk.table.height = 0.08,
                      cumevents.height = 0.08,
                      cumcensor.height = 0.08,
                      xlab = 'Days',
                      ylab = 'Cumulative proportion\n',
                      ylim = c(0, 1),
                      palette = c('#000000'),
                      legend = 'none',
                      ggtheme = theme_new())

## Save
ggsave(filename = 'figures/survival-curve.png',
       plot = print(gg_plot),
       width = 12.7,
       height = 12)
```

Plot a basic KM curve conditioned on painful SN

In addition we performed a log-rank test to asssess for differences between painful and non-painful SN strata.

```
# Incidence ~ pain
km_pain <- survfit(Surv(time = approximate_day,
                         event = hivsn_coded) ~ pain,
                    data = data_surv)

# Summary
km_pain
```

```
## Call: survfit(formula = Surv(time = approximate_day, event = hivsn_coded) ~
##     pain, data = data_surv)
##
##            n events median 0.95LCL 0.95UCL
## pain=no   96     11     NA      NA      NA
## pain=yes 24      9     NA      84      NA
```

```
summary(km_pain)
```

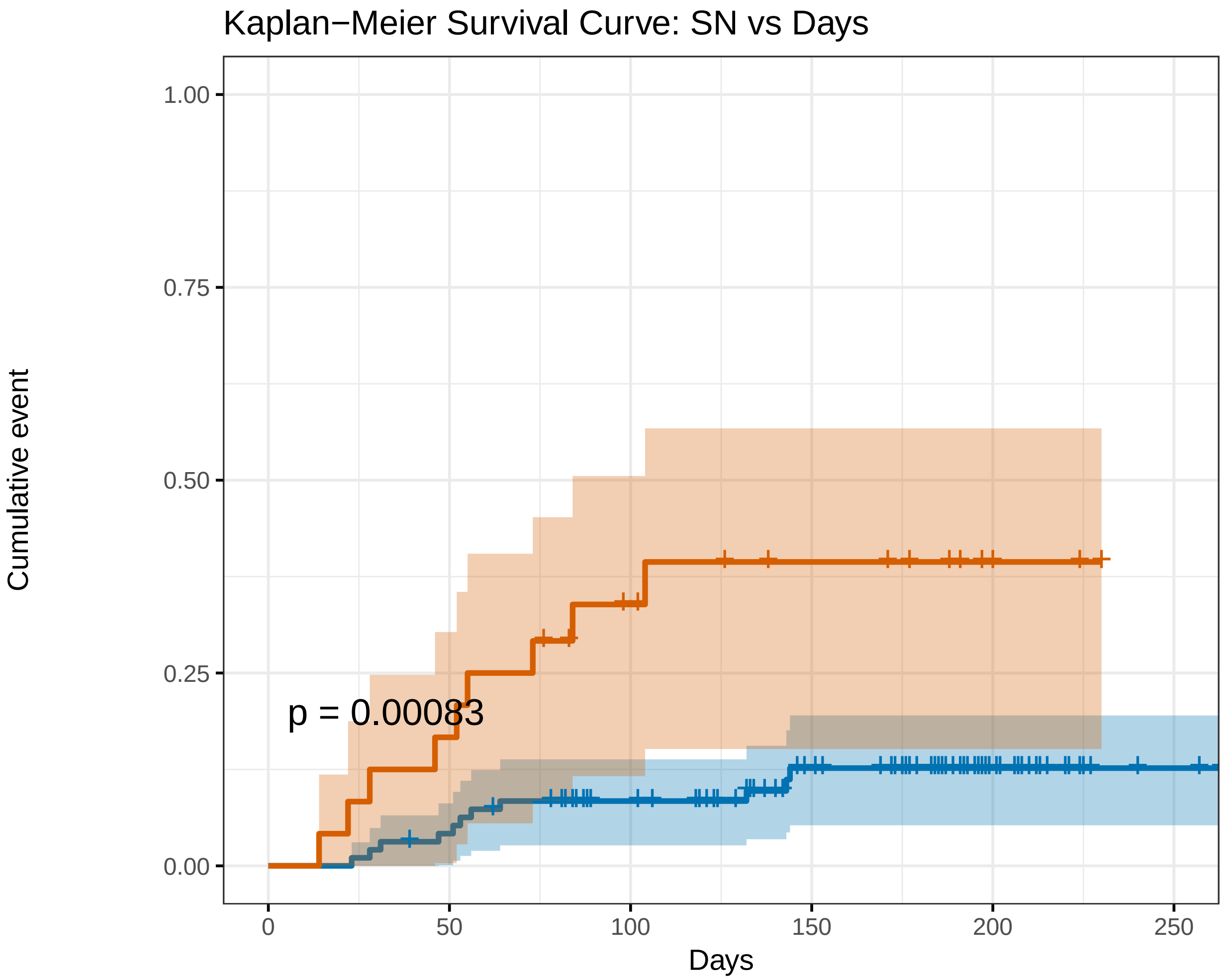
```
## Call: survfit(formula = Surv(time = approximate_day, event = hivsn_coded) ~
##     pain, data = data_surv)
##
##                 pain=no
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##    23     96       1    0.990  0.0104        0.969        1.000
##    28     95       1    0.979  0.0146        0.951        1.000
##    31     94       1    0.969  0.0178        0.935        1.000
##    47     92       1    0.958  0.0205        0.919        0.999
##    51     91       1    0.948  0.0228        0.904        0.993
##    53     90       1    0.937  0.0248        0.890        0.987
##    56     89       1    0.927  0.0267        0.876        0.980
##    64     87       1    0.916  0.0284        0.862        0.973
##   132     70       1    0.903  0.0309        0.844        0.966
##   143     61       1    0.888  0.0337        0.824        0.957
##   144     60       1    0.873  0.0363        0.805        0.947
##
##                 pain=yes
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##    14     24       1    0.958  0.0408        0.882        1.000
##    22     23       1    0.917  0.0564        0.813        1.000
##    28     22       1    0.875  0.0675        0.752        1.000
##    46     21       1    0.833  0.0761        0.697        0.997
##    52     20       1    0.792  0.0829        0.645        0.972
##    55     19       1    0.750  0.0884        0.595        0.945
##    73     18       1    0.708  0.0928        0.548        0.916
##    84     15       1    0.661  0.0979        0.495        0.884
##   104     12       1    0.606  0.1041        0.433        0.849
```

```
# Log-rank test
survdiff(Surv(time = visit_day,
              event = hivsn_coded) ~ pain,
         data = data_surv)
```

```
## Call:
## survdiff(formula = Surv(time = visit_day, event = hivsn_coded) ~
##     pain, data = data_surv)
##
##            N Observed Expected (O-E)^2/E (O-E)^2/V
## pain=no   96       11    16.89      2.05      13.4
## pain=yes 24        9     3.11     11.13      13.4
```

```
##
##  Chisq= 13.4  on 1 degrees of freedom, p= 3e-04
```

```
# Plot
ggsurvplot(km_pain,
           conf.int = TRUE,
           fun = 'event',
           risk.table = 'abs_pct',
           cumcensor = TRUE,
           cumevents = TRUE,
           surv.median.line = 'hv',
           pval = TRUE,
           ggtheme = theme_bw(),
           tables.theme = theme(plot.title = element_text(size = 12,
                                                        face = 'bold'),
                                panel.border = element_blank(),
                                panel.grid.major.x = element_blank(),
                                axis.title = element_blank(),
                                axis.text.x = element_blank(),
                                axis.ticks = element_blank()),
           fontsize = 4,
           tables.height = 0.14,
           legend = 'none',
           legend.labs = c('Non-painful SN', 'Painful SN'),
           palette = c('#0072B2', '#D55E00'),
           ylim = c(0, 1),
           xlab = 'Days',
           title = 'Kaplan-Meier Survival Curve: SN vs Days')
```



Number at risk: n (%)

Non−painful SN	96 (100)	91 (95)	78 (81)	57 (59)	22 (23)	2 (2)
Painful SN	24 (100)	20 (83)	13 (54)	9 (38)	3 (12)	0 (0)

Cumulative number of events

Non−painful SN	0	4	8	11	11	11
Painful SN	0	4	8	9	9	9

Cumulative number of censoring

Non−painful SN	0	1	10	28	63	83
Painful SN	0	0	3	6	13	15

Multivariable modeling

In an exploratory analysis using Cox proportional hazard models (not shown here) various predictors violated assumptions of the model (e.g., proportional hazard, linearity, no influence points). Therefore we used logistic regression modelling, with visit 1 charateristics as predictors of SN onset.

Model data

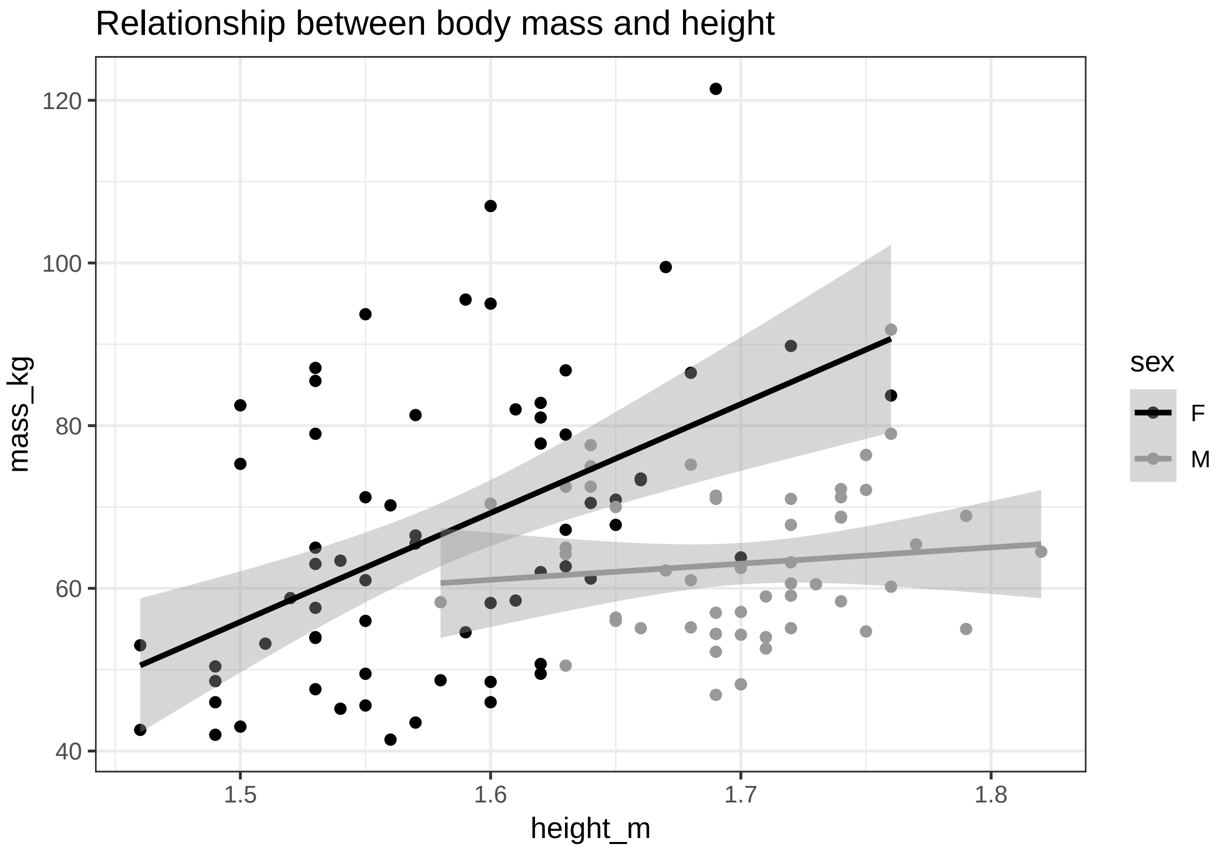
The model does not include:

1. `diabetes_hba1c` because nobody had HbA1c $>$ 7%.
2. `vitaminB12_deficiency` because only one individual had a deficiency.
3. `mass_kg` because it is likely to be co-linear with `height_m`.
4. `consumes_alcohol` because this information was deemed to be encoded in `alcohol_units.week`.

Model selection

Check height vs mass assumption

```
# Double-check presumed 'mass_kg' vs 'height_m' relationship before proceeding
ggplot(data = data_model) +
    aes(x = height_m,
        y = mass_kg,
        colour = sex) +
    geom_point() +
    geom_smooth(method = 'lm') +
    scale_colour_manual(values = c('#000000', '#999999')) +
    labs(title = 'Relationship between body mass and height') +
    theme_bw()
```



```
# Males
with(data_model[data_model$sex == 'M', ], cor.test(height_m, mass_kg))
```

```
##
##  Pearson's product-moment correlation
##
## data:  height_m and mass_kg
## t = 0.77427, df = 52, p-value = 0.4423
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.1657403  0.3641103
## sample estimates:
##       cor
## 0.1067581
```

```
# Females
with(data_model[data_model$sex == 'F', ], cor.test(height_m, mass_kg))
```

```
##
##  Pearson's product-moment correlation
##
## data:  height_m and mass_kg
## t = 4.42, df = 64, p-value = 3.899e-05
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.2735853 0.6495961
## sample estimates:
##      cor
## 0.483596
```

The two variables are related in females, but not in males. We decided to omit `weight_kg` in favour of `height_m` because of this relationship, and the stronger historical association between height and SN.

Elastic net regression

We chose to use elastic net for variable selection. Elastic net is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the statistical model it produces.

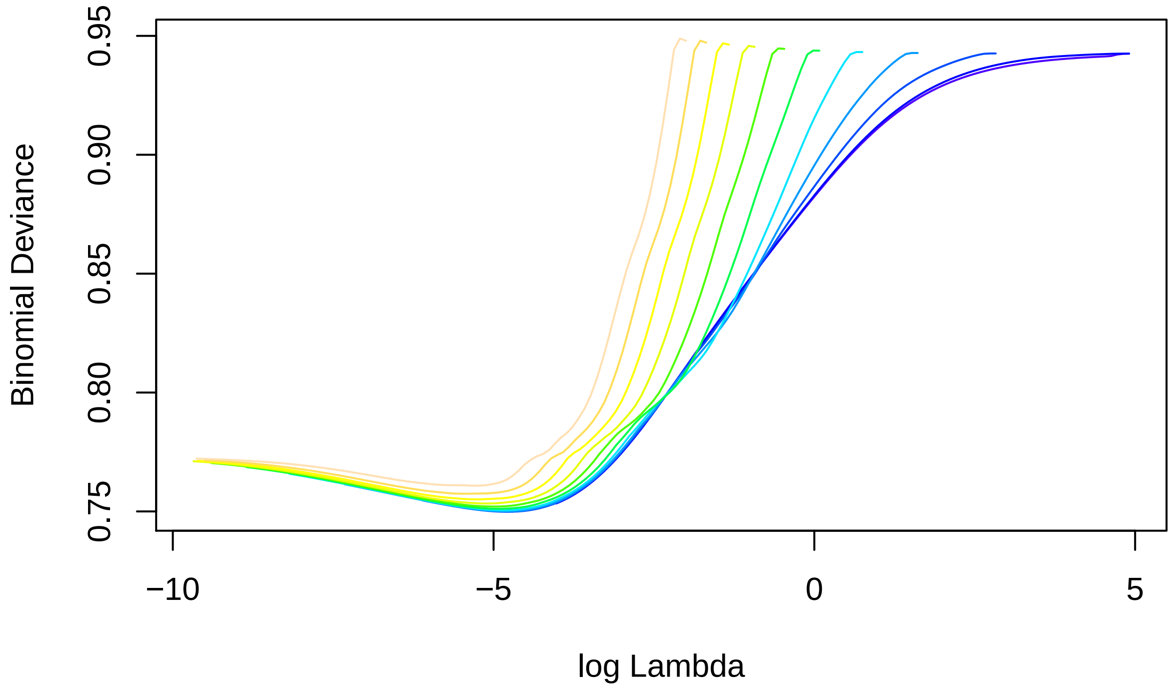
The process involves performing a 10-fold cross validation to find the optimal *lambda* (penalization parameter). And then running the analysis and extracting the model based on the best *lambda*.

Cross-validation to estimate alpha and lambda

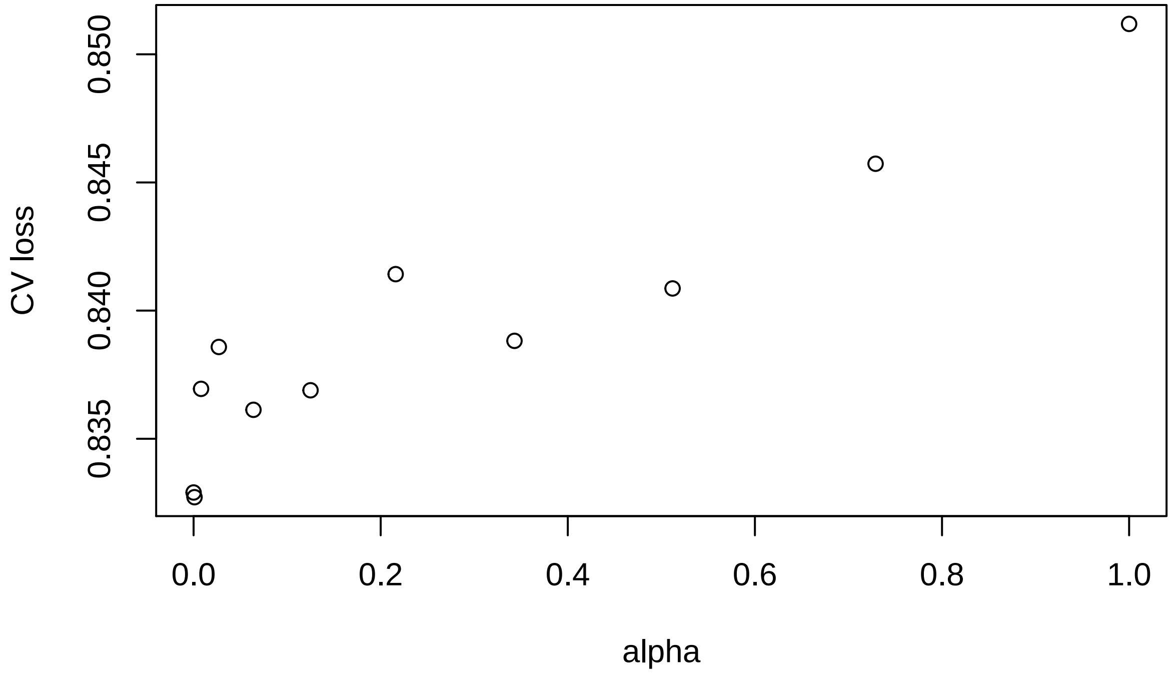
```
# Remove excluded varibales
data_model %<>%
    select(-mass_kg)

cvafit <- cva.glmnet(sn ~ .,
                     data = data_model,
                     family = 'binomial')

# Print CVA object
print(cvafit)
```

```
## Call:
## cva.glmnet.formula(formula = sn ~ ., data = data_model, family = "binomial")
##
## Model fitting options:
##     Sparse model matrix: FALSE
##     Use model.frame: FALSE
##     Alpha values: 0 0.001 0.008 0.027 0.064 0.125 0.216 0.343 0.512 0.729 1
##     Number of crossvalidation folds for lambda: 10
```

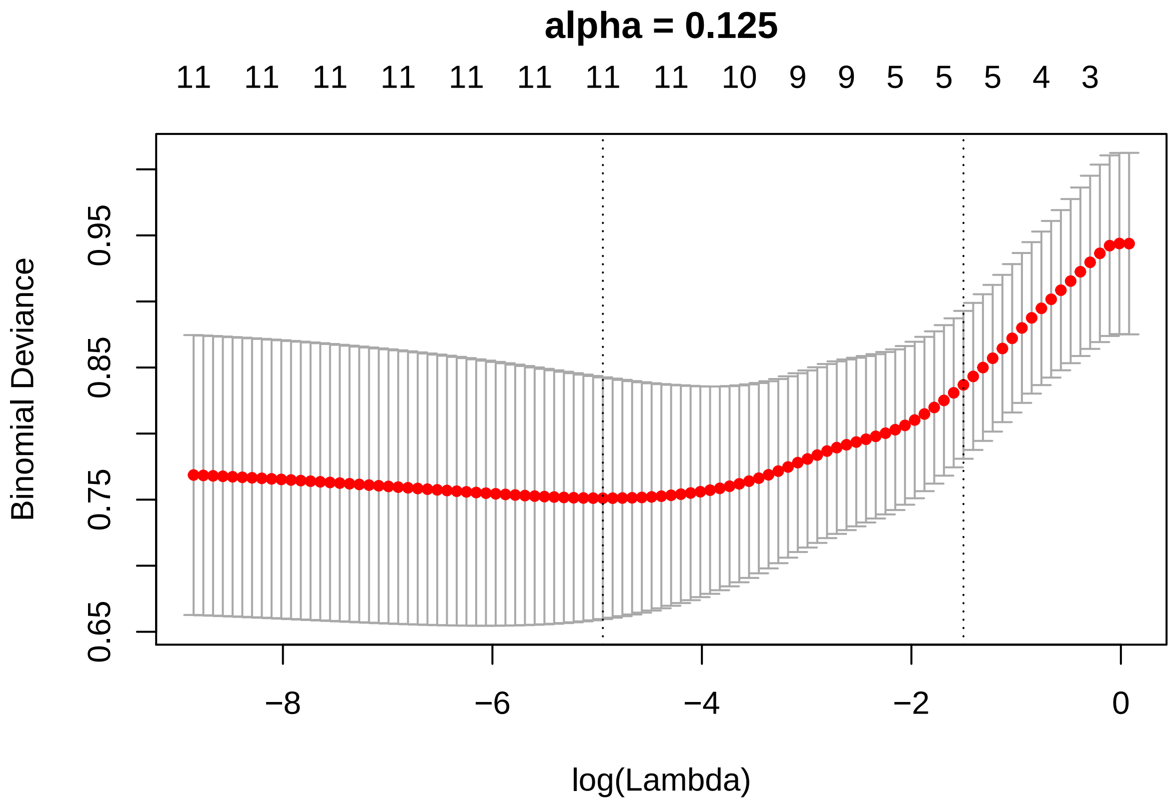
```
# Plot CVA object log(lambda) vs CV loss (deviance) for each value of alpha
plot(cvafit)
```



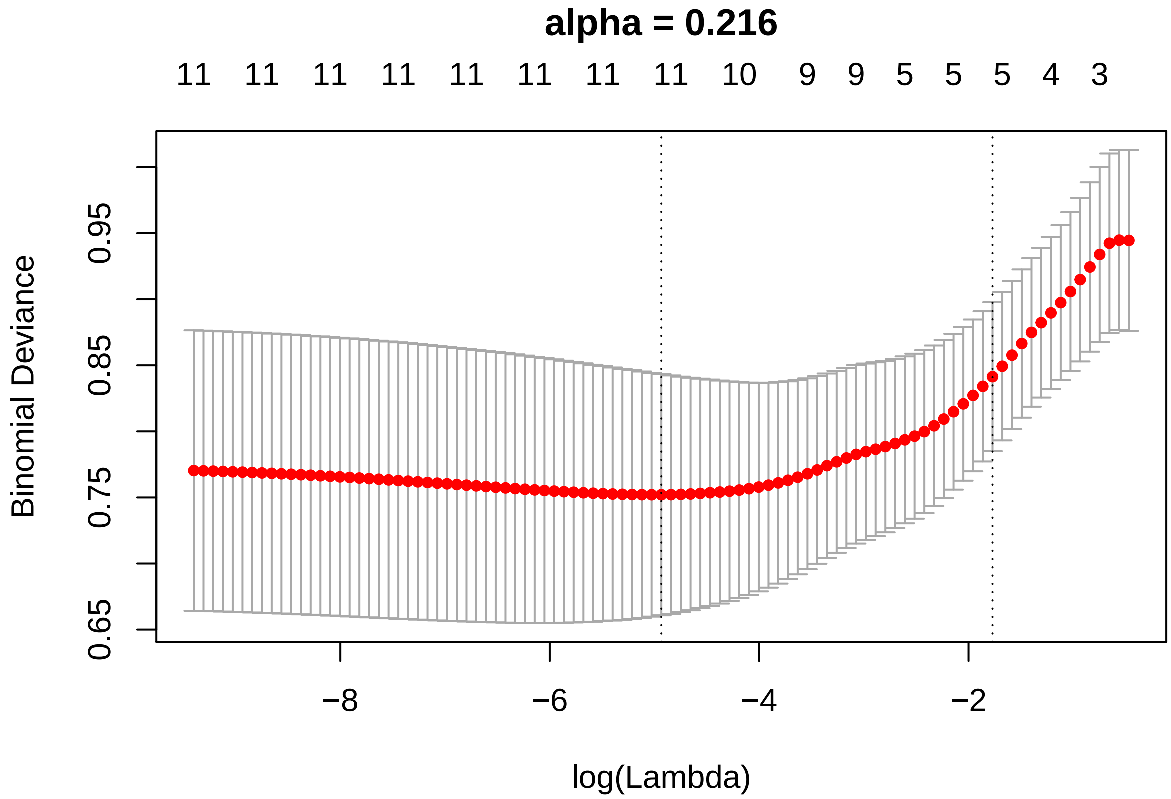
```
# Plot minimum CV loss (deviance) for each value of alpha
minlossplot(cvafit)
```



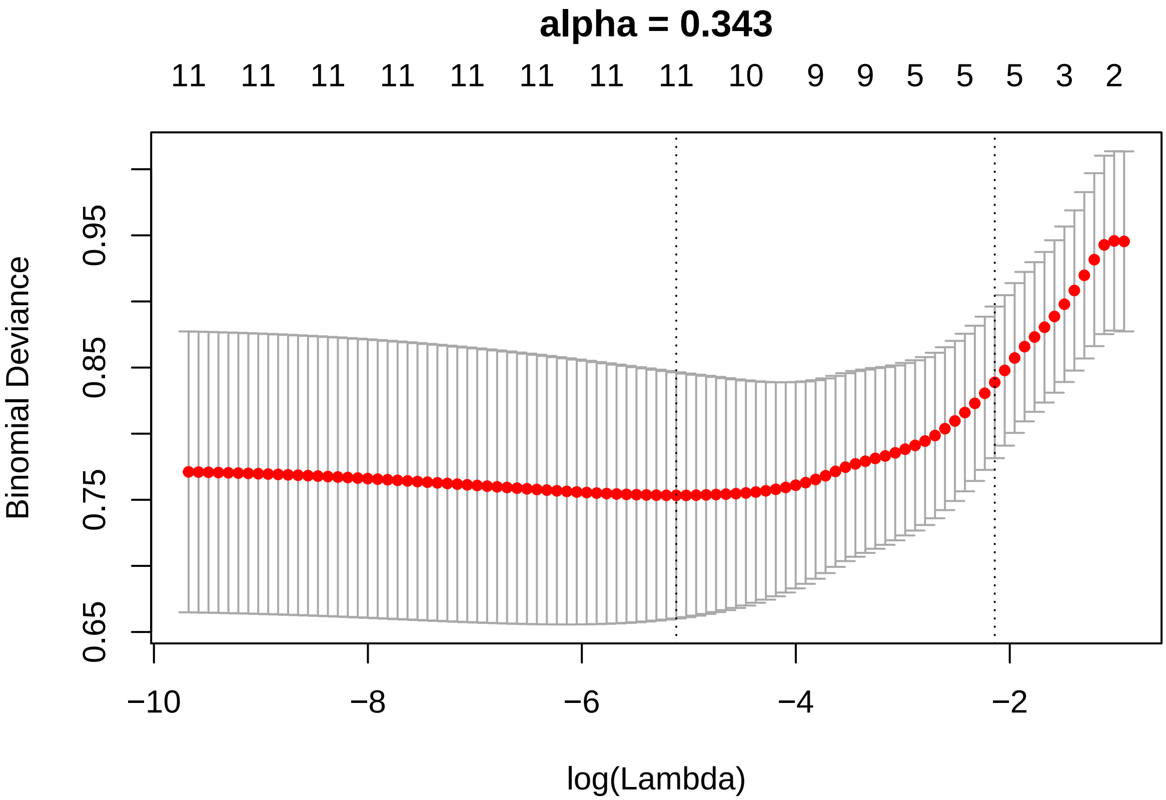
```
# Print model at each of the three lowest CV loss alphas
## alpha = 0.125
plot(cvafit$modlist[[6]])
title('alpha = 0.125', line = 2.5)
```



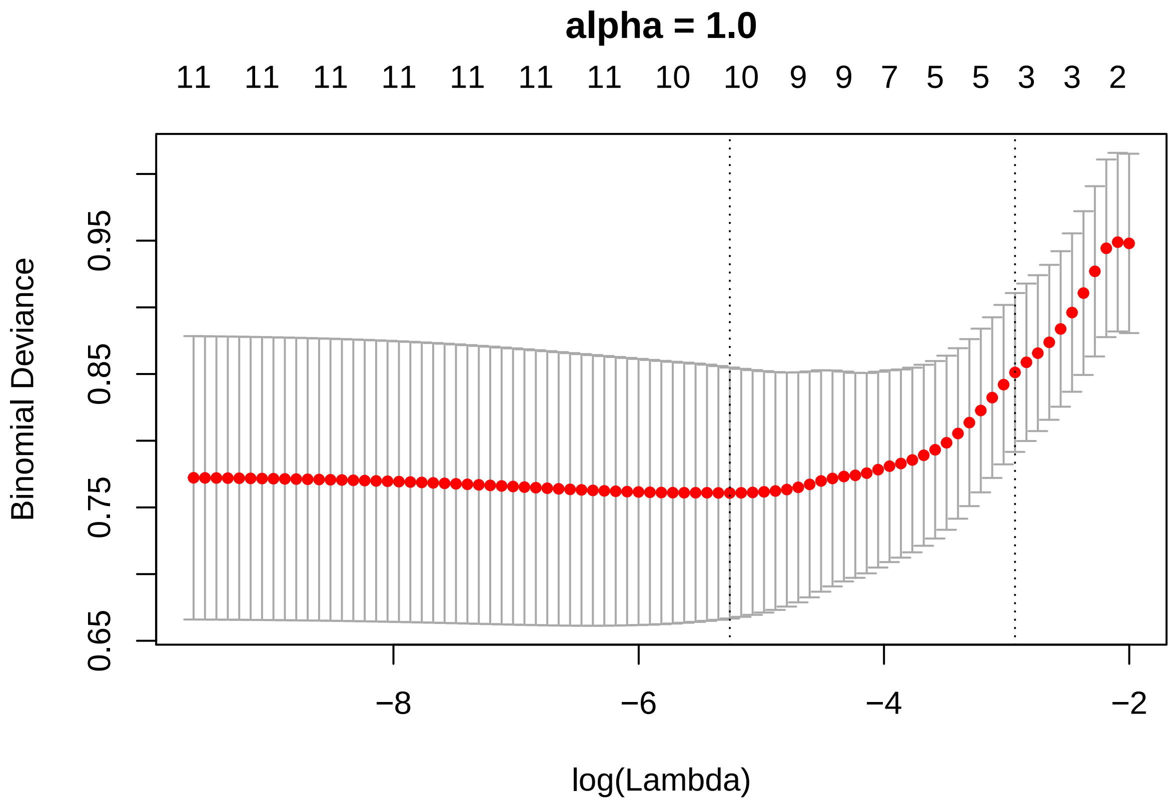
```
## alpha = 0.216
plot(cvafit$modlist[[7]])
title('alpha = 0.216', line = 2.5)
```



```
## alpha = 0.343
plot(cvafit$modlist[[8]])
title('alpha = 0.343', line = 2.5)
```



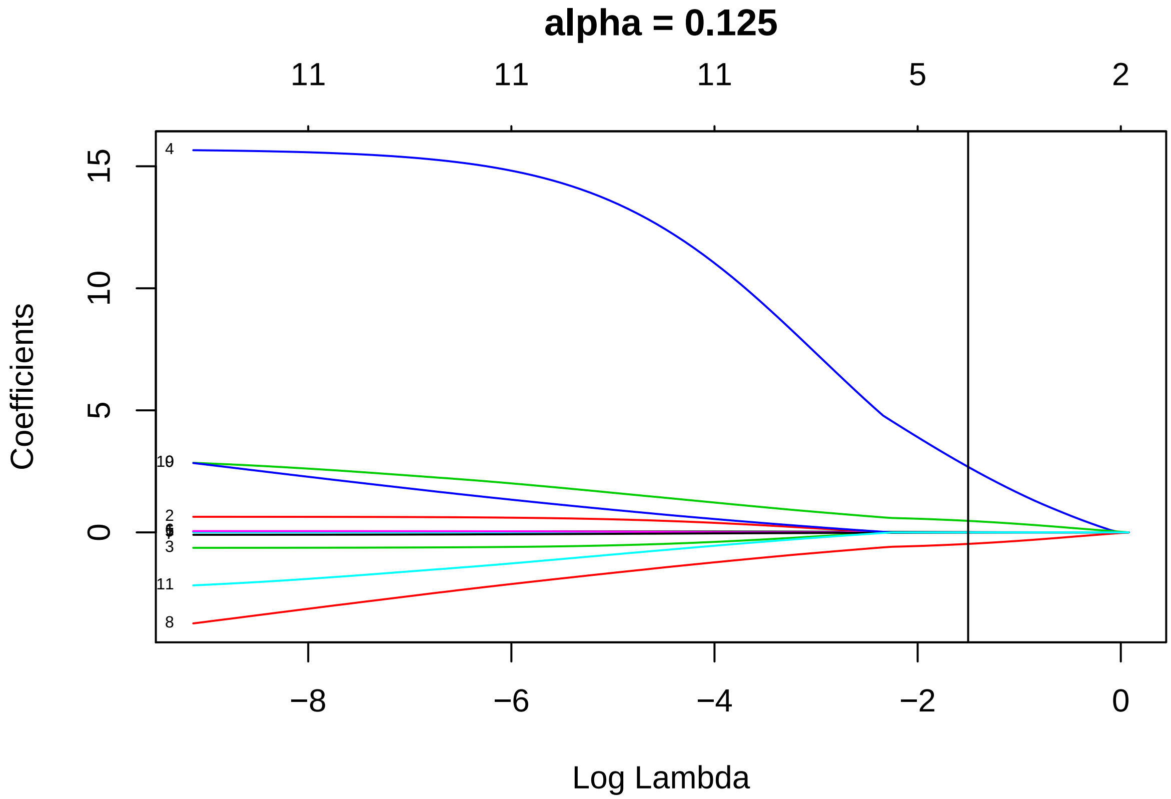
```
## and alpha = 1
plot(cvafit$modlist[[11]])
title('alpha = 1.0', line = 2.5)
```



```
# Extract best lambda for each alpha (lambda.1se)
## alpha = 0.125
a125 <- cvafit$modlist[[6]]$lambda.1se
## alpha = 0.216
a216 <- cvafit$modlist[[7]]$lambda.1se
## alpha = 0.343
a343 <- cvafit$modlist[[8]]$lambda.1se
## and alpha = 1
a100 <- cvafit$modlist[[11]]$lambda.1se
```

Fit the models using CV alphas and best lambdas

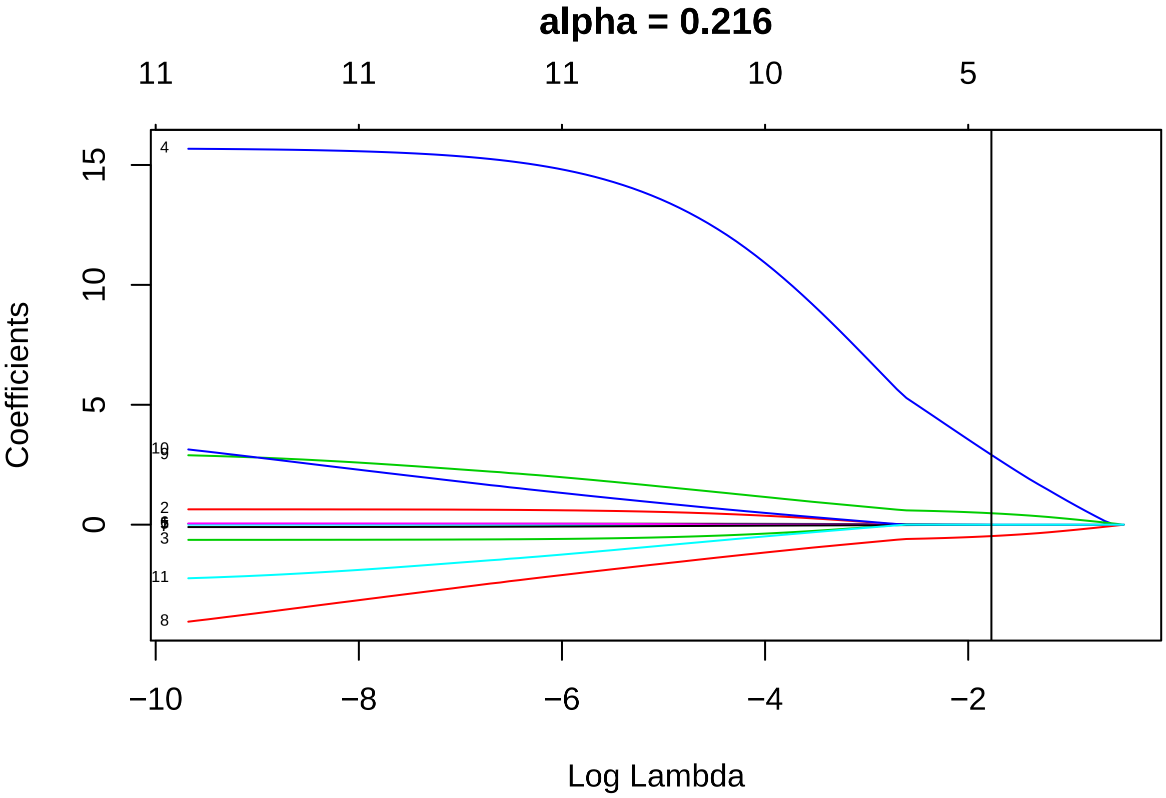
```
# Fit the models for each alpha
## alpha = 0.125
model_a125 <- glmnet(sn ~ .,
                     data = data_model,
                     alpha = 0.125,
                     family = 'binomial')

## alpha = 0.216
model_a216 <- glmnet(sn ~ .,
                     data = data_model,
                     alpha = 0.216,
                     family = 'binomial')

## alpha = 0.343
model_a343 <- glmnet(sn ~ .,
                     data = data_model,
                     alpha = 0.343,
                     family = 'binomial')

## alpha = 1.0
model_a100 <- glmnet(sn ~ .,
                     data = data_model,
                     alpha = 1,
                     family = 'binomial')

# Plot and get coefficients for each model
## alpha = 0.125
plot(model_a125,
     xvar = 'lambda',
     label = TRUE)
abline(v = log(a125))
title('alpha = 0.125', line = 2.5)
```



```
coef(model_a125, s = a125)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                                  1
## (Intercept)           -6.006264858
## age_years              0.009890009
## sexF                   .
## sexM                   .
## height_m               2.683933491
## CD4_cell.ul            .
## viral_load_copies.ml   .
## alcohol_units.week    -0.005570078
## TB_currentno          -0.471945279
## TB_currentyes          0.471974552
## rifafour_treatmentno   .
## rifafour_treatmentyes  .
```

```
## alpha = 0.216
plot(model_a216,
     xvar = 'lambda',
     label = TRUE)
abline(v = log(a216))
title('alpha = 0.216', line = 2.5)
```

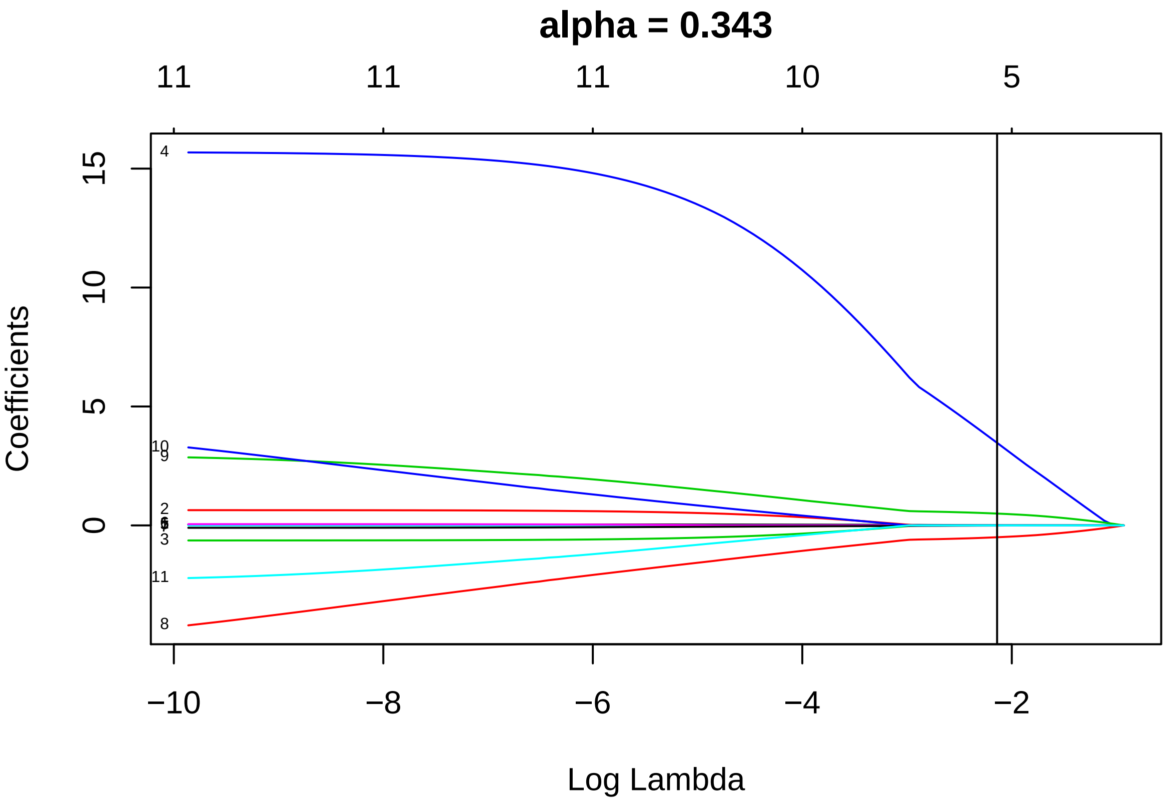


```
coef(model_a216, s = a216)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                                  1
## (Intercept)           -6.319823675
## age_years              0.008346678
## sexF                   .
## sexM                   .
## height_m               2.909586666
## CD4_cell.ul            .
## viral_load_copies.ml   .
## alcohol_units.week    -0.004767905
## TB_currentno          -0.476746473
## TB_currentyes          0.476771526
## rifafour_treatmentno   .
## rifafour_treatmentyes  .
```

```
## alpha = 0.343
```

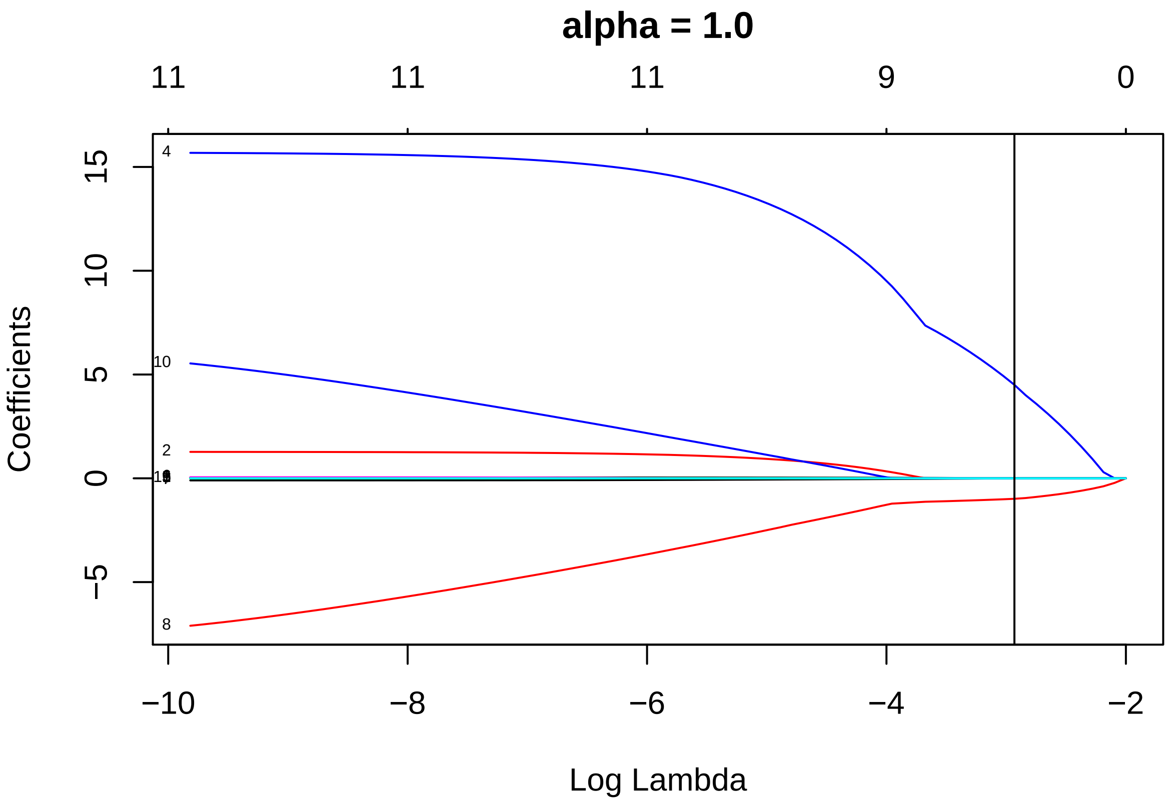
```r
plot(model_a343,
     xvar = 'lambda',
     label = TRUE)
abline(v = log(a343))
title('alpha = 0.343', line = 2.5)
```



```r
coef(model_a343, s = a343)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                                 1
## (Intercept)          -7.219271003
## age_years             0.007976664
## sexF                  .
## sexM                  .
## height_m              3.470945470
## CD4_cell.ul           .
## viral_load_copies.ml  .
## alcohol_units.week   -0.005248733
## TB_currentno         -0.498112450
## TB_currentyes         0.497577527
## rifafour_treatmentno  .
## rifafour_treatmentyes .
```

```r
## alpha = 1.0
plot(model_a100,
     xvar = 'lambda',
     label = TRUE)
abline(v = log(a100))
title('alpha = 1.0', line = 2.5)
```



```
coef(model_a100, s = a100)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                                    1
## (Intercept)           -8.156213e+00
## age_years              4.256286e-04
## sexF                   .
## sexM                   .
## height_m               4.502745e+00
## CD4_cell.ul            .
## viral_load_copies.ml   .
## alcohol_units.week    -2.535917e-03
## TB_currentno          -9.870485e-01
## TB_currentyes          3.991934e-14
## rifafour_treatmentno   .
## rifafour_treatmentyes  .
```

Across all alphas, and best lambdas, the output shows the best model includes `height_m` and `TB_current`.

Session information

```
sessionInfo()
```

```
## R version 3.6.0 (2019-04-26)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Mojave 10.14.4
##
```

```
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] survminer_0.4.3   ggpubr_0.2        survival_2.44-1.1
##  [4] boot_1.3-22       glmnetUtils_1.1.2 lubridate_1.7.4
##  [7] forcats_0.4.0     stringr_1.4.0     dplyr_0.8.0.1
## [10] purrr_0.3.2       readr_1.3.1       tidyr_0.8.3
## [13] tibble_2.1.1      ggplot2_3.1.1     tidyverse_1.2.1
## [16] magrittr_1.5
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.1        lattice_0.20-38   zoo_1.8-5
##  [4] utf8_1.1.4        assertthat_0.2.1  glmnet_2.0-16
##  [7] digest_0.6.18     foreach_1.4.4     R6_2.4.0
## [10] cellranger_1.1.0  plyr_1.8.4        backports_1.1.4
## [13] evaluate_0.13     httr_1.4.0        pillar_1.3.1
## [16] rlang_0.3.4       lazyeval_0.2.2    readxl_1.3.1
## [19] rstudioapi_0.10   data.table_1.12.2 Matrix_1.2-17
## [22] rmarkdown_1.12    labeling_0.3      splines_3.6.0
## [25] munsell_0.5.0     broom_0.5.2       compiler_3.6.0
## [28] modelr_0.1.4      xfun_0.6          pkgconfig_2.0.2
## [31] htmltools_0.3.6   tidyselect_0.2.5  gridExtra_2.3
## [34] km.ci_0.5-2       codetools_0.2-16  fansi_0.4.0
## [37] crayon_1.3.4      withr_2.1.2.9000  grid_3.6.0
## [40] nlme_3.1-139      jsonlite_1.6      xtable_1.8-4
## [43] gtable_0.3.0      KMsurv_0.1-5      scales_1.0.0
## [46] cli_1.1.0         stringi_1.4.3     xml2_1.2.0
## [49] survMisc_0.5.5    generics_0.0.2    iterators_1.0.10
## [52] tools_3.6.0       cmprsk_2.2-7      glue_1.3.1
## [55] hms_0.4.2         parallel_3.6.0    yaml_2.2.0
## [58] colorspace_1.4-1  rvest_0.3.3       knitr_1.22
## [61] haven_2.1.0
```