Utilizing VitalDBR to determine the pulse pressure variation's influence on mortality of open surgery patients

```
library(devtools)
```

```
## Loading required package: usethis
```

```
install_github('legendenomgeorg/VitalDBR/VitalDBR')
```

```
## Skipping install of 'VitalDBR' from a github remote, the SHA1 (cb8f319f) has not changed since last i
##   Use `force = TRUE` to force installation
```

```
library(VitalDBR)
library(stats)
library(waveformtools)
library(ggplot2)
library(mgcv)
```

```
## Loading required package: nlme
```

```
## This is mgcv 1.8-38. For overview type 'help("mgcv-package")'.
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:nlme':
##
##     collapse
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(patchwork)
```

```
data_art <- VitalDBR::load_case('SNUADC/ART', 1)
data_awp <- VitalDBR::load_case('Primus/AWP', 1)
```

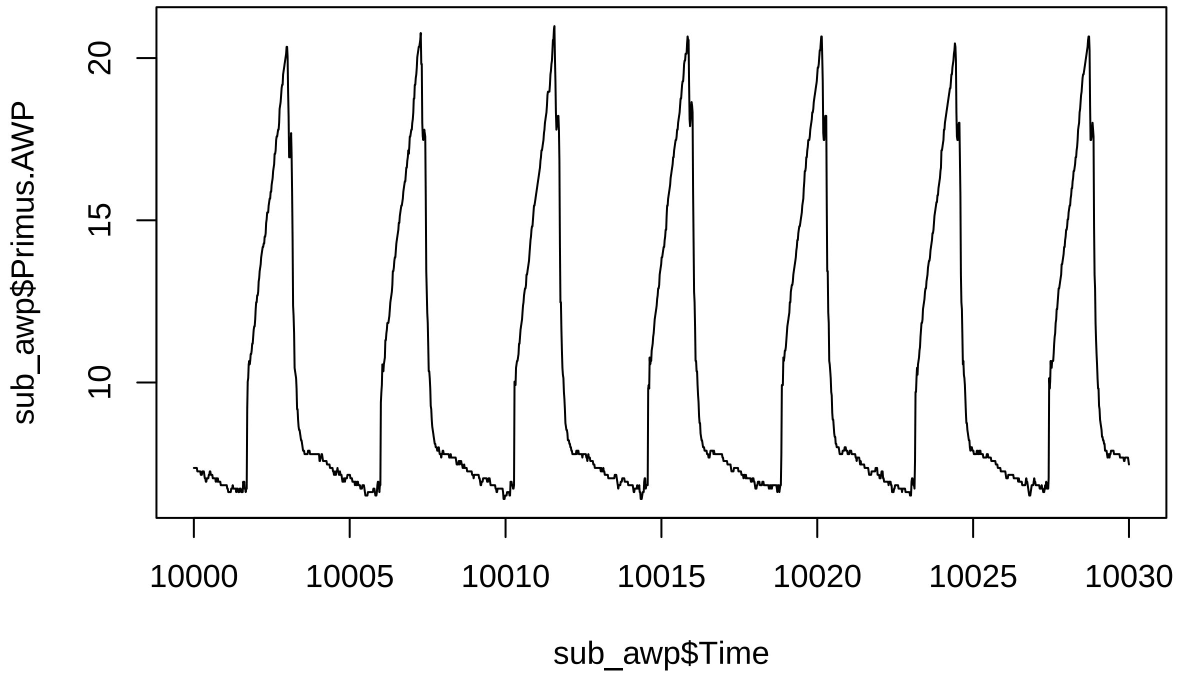
```
start = 10000 # Starter fra sekund "start"
sec = 30 # Varer "sec" sekunder
```

Subset AWP time series

```
sub_awp <- VitalDBR::subset_data(data = data_awp, seconds = sec, start_sec = start)
head(sub_awp)
```

```
##       Time Primus.AWP
## 1 10000.00     7.3664
## 2 10000.02     7.3664
## 3 10000.03     7.3664
## 4 10000.05     7.3664
## 5 10000.06     7.3664
## 6 10000.08     7.3664
```

```
plot(sub_awp$Time, sub_awp$Primus.AWP, type='l')
```



```
insp_start <- VitalDBR::get_inspiration_start(sub_awp)
head(insp_start)
```

```
##       time
## 1 10001.65
## 2 10005.94
## 3 10010.22
## 4 10014.51
## 5 10018.78
## 6 10023.09
```

```
sub_art <- VitalDBR::subset_data(data = data_art, seconds = sec, start_sec = start, filter=TRUE)
waveformtools::dygraph_signal(sub_art, SNUADC.ART, SNUADC.ART_filt)
```

```
## PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installed, pleas
```

ABP beats

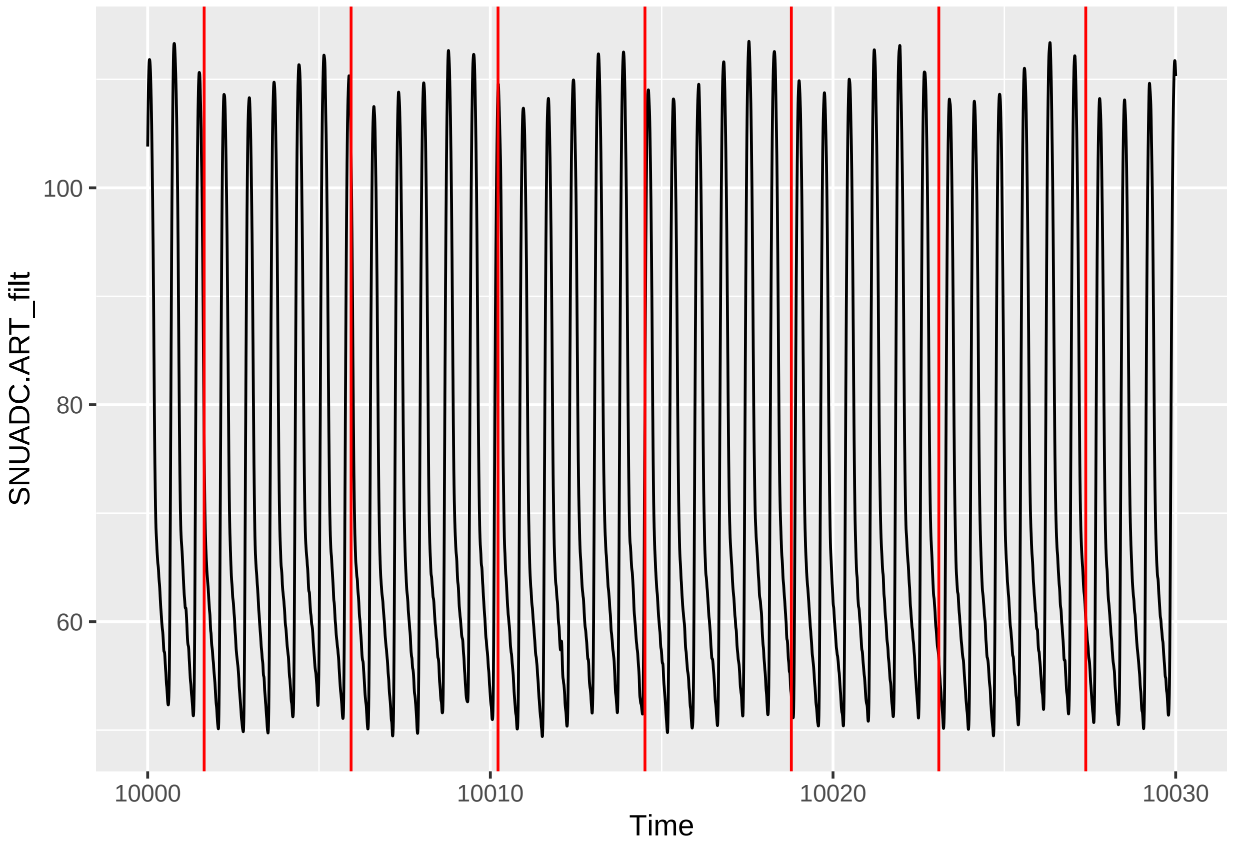
```
beats <- waveformtools::find_abp_beats(sub_art,3,1)[-1,] #skipper lige første row, fordi der ikke er no
```

```
## Sample rate not set. Returning beat length in samples
```

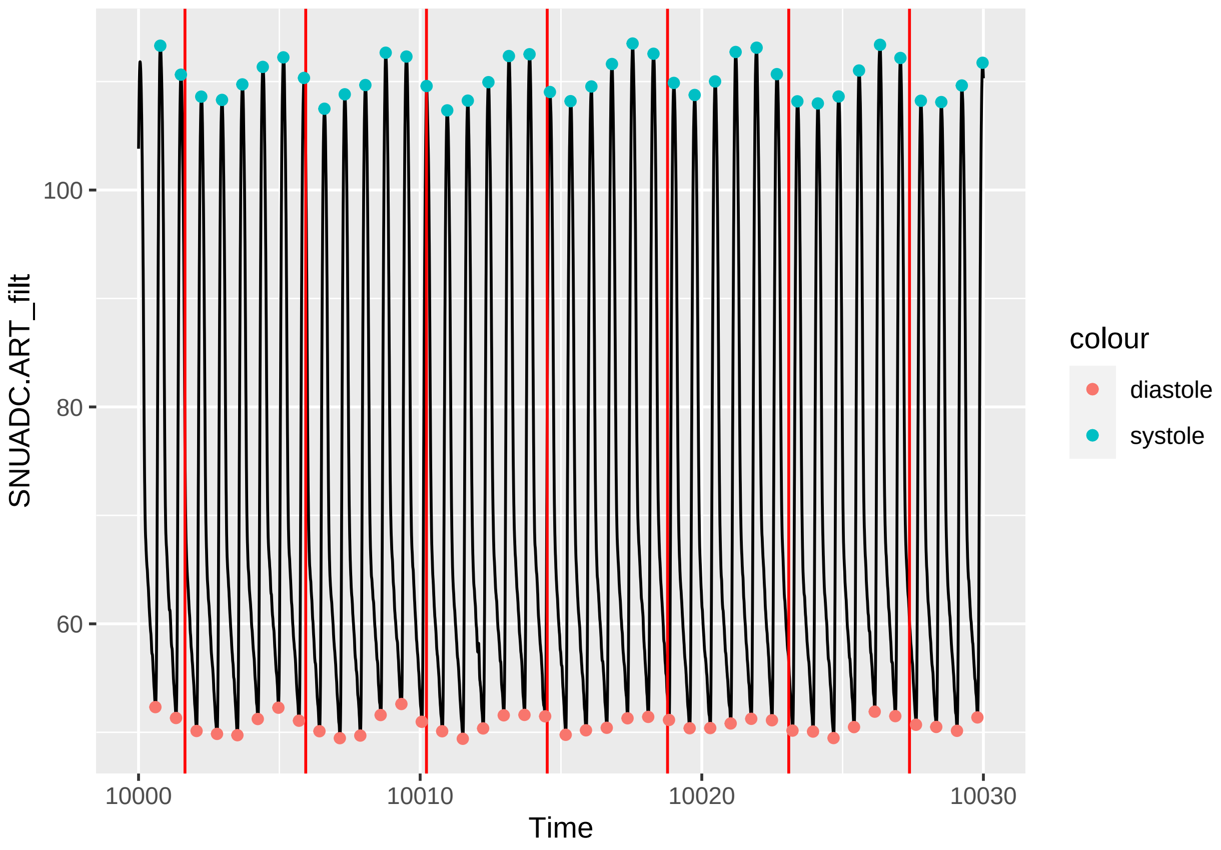
Johannes plots

```
abp_plot <- ggplot(sub_art, aes(Time, SNUADC.ART_filt)) +
geom_line() +
```

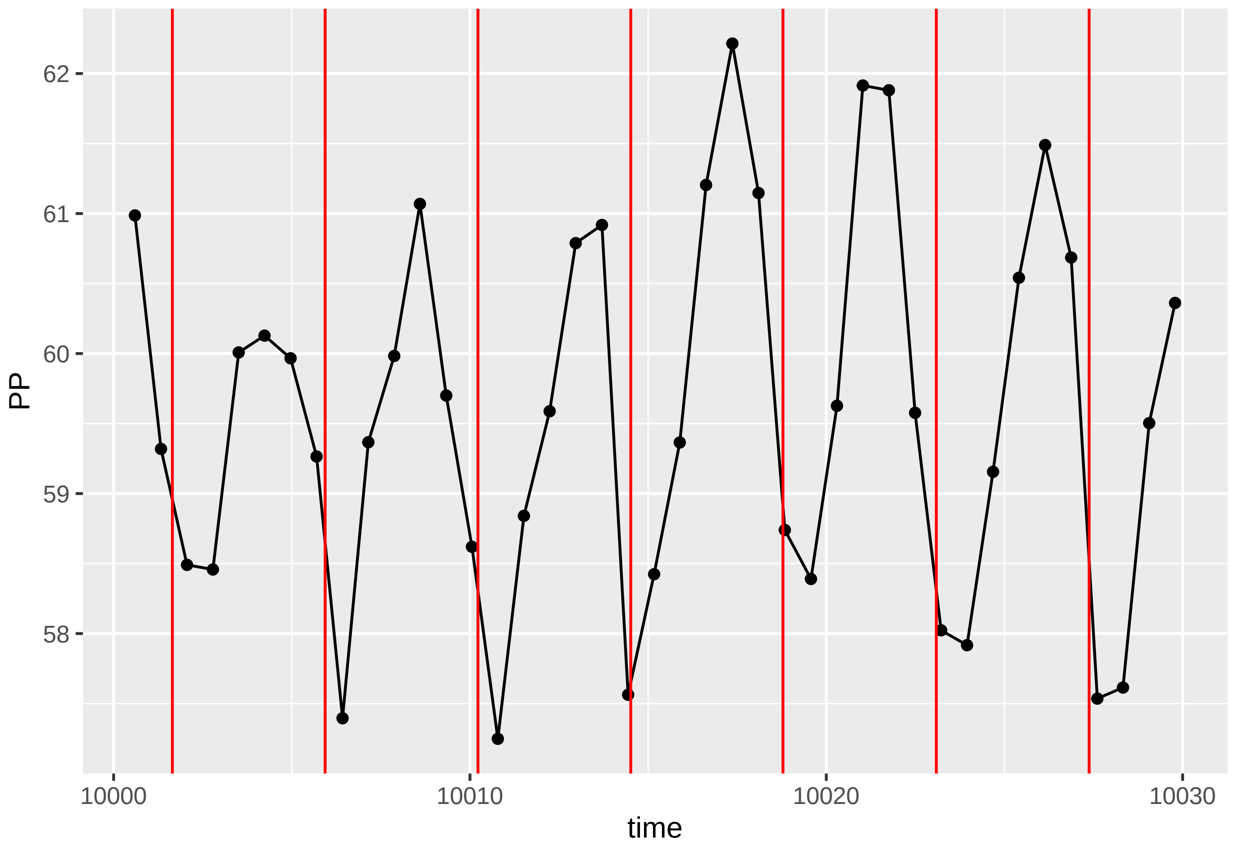
```r
geom_vline(aes(xintercept = time), color = "red",
data = insp_start)
abp_plot
```



```r
abp_plot +
  geom_point(aes(x = time,
    y = dia,
    colour = "diastole"),
    data = beats) +
  geom_point(aes(x = time_systole,
    y = sys,
    colour = "systole"),
    data = beats)
```



```
pp_plot <- ggplot(beats, aes(time, PP)) +
  geom_line() +
  geom_point() +
  geom_vline(aes(xintercept = time), color = "red",
  data = insp_start)
pp_plot
```



```
head(beats)
```

```
## # A tibble: 6 x 10
##     time   dia   sys    PP beat_len time_systole dia_pos sys_pos
##    <dbl> <dbl> <dbl> <dbl>    <int>        <dbl>   <dbl>   <dbl>
## 1 10001.  52.3  113.  61.0      366       10001.     300     387
## 2 10001.  51.3  111.  59.3      365       10002.     666     753
## 3 10002.  50.1  109.  58.5      363       10002.    1031    1114
## 4 10003.  49.9  108.  58.5      362       10003.    1394    1483
## 5 10004.  49.7  110.  60.0      362       10004.    1756    1844
## 6 10004.  51.2  111.  60.1      366       10004.    2118    2207
## # ... with 2 more variables: .noise_wiggliness <dbl>,
## #   .noise_pos_after_sys <dbl>
```

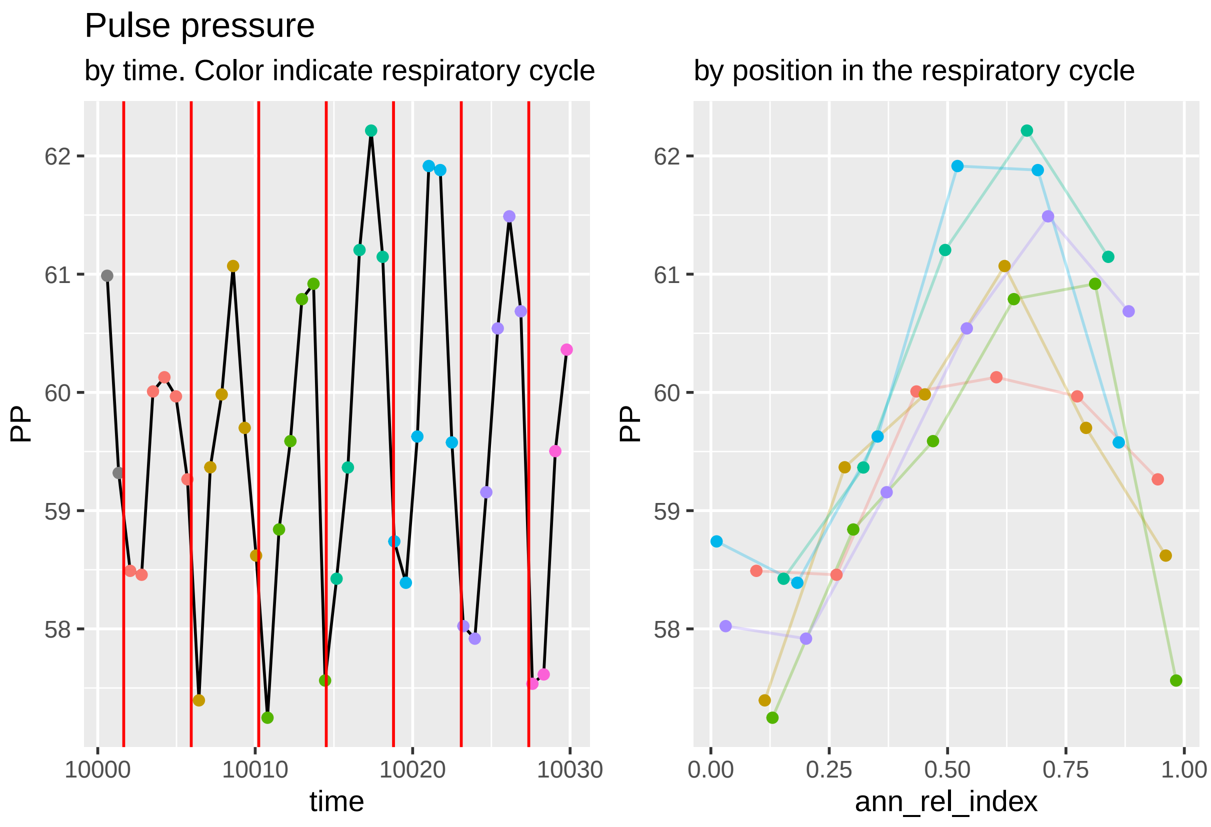
```
beats_indexed <- waveformtools::add_time_since_event(beats, time_event = insp_start$time)
# ann_rel_index starter forfra ved hver inspiration.
head(beats_indexed)
```

```
## # A tibble: 6 x 14
##     time   dia   sys    PP beat_len time_systole dia_pos sys_pos
##    <dbl> <dbl> <dbl> <dbl>    <int>        <dbl>   <dbl>   <dbl>
## 1 10001.  52.3  113.  61.0      366       10001.     300     387
## 2 10001.  51.3  111.  59.3      365       10002.     666     753
## 3 10002.  50.1  109.  58.5      363       10002.    1031    1114
## 4 10003.  49.9  108.  58.5      362       10003.    1394    1483
## 5 10004.  49.7  110.  60.0      362       10004.    1756    1844
## 6 10004.  51.2  111.  60.1      366       10004.    2118    2207
## # ... with 6 more variables: .noise_wiggliness <dbl>,
## #   .noise_pos_after_sys <dbl>, ann_index <dbl>, ann_n <int>,
## #   ann_cycle_len <dbl>, ann_rel_index <dbl>
```

```
pp_plot_color <- ggplot(beats_indexed, aes(time, PP)) +
geom_line() +
# insp_n is a unique (consecutive) number for each respiratory cycle
geom_point(aes(color = as.factor(ann_n)), show.legend = FALSE) +
geom_vline(aes(xintercept = time), color = "red",
data = insp_start) +
labs(title = "Pulse pressure",
subtitle = "by time. Color indicate respiratory cycle")
pp_insp_plot <- ggplot(beats_indexed,
aes(
ann_rel_index,
PP,
group = as.factor(ann_n),
color = as.factor(ann_n)
)
) +
geom_line(alpha = 0.3, show.legend = FALSE) +
# insp_n is a unique (consecutive) number for each respiratory cycle
geom_point(aes(color = as.factor(ann_n)), show.legend = FALSE) +
labs(subtitle = "by position in the respiratory cycle")
pp_plot_color + pp_insp_plot
```

```
## Warning: Removed 6 row(s) containing missing values (geom_path).
```

```
## Warning: Removed 6 rows containing missing values (geom_point).
```

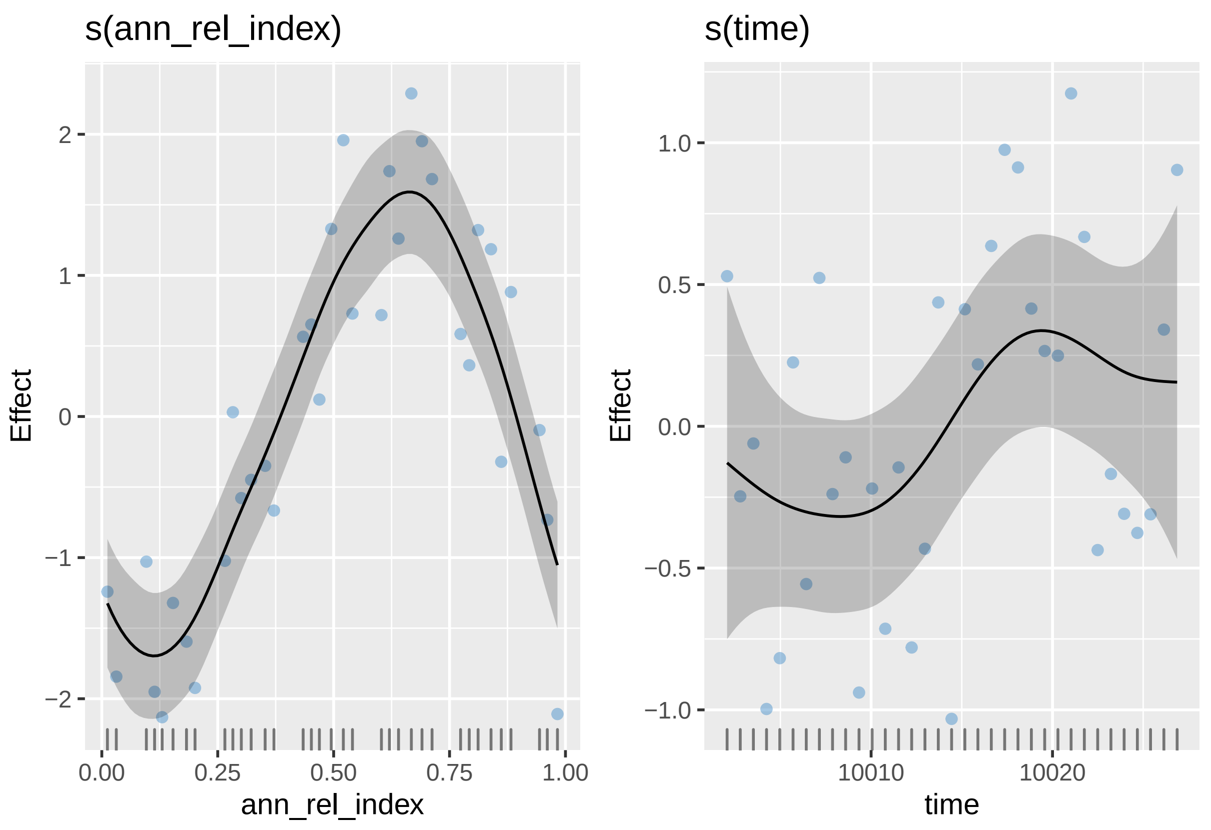


```
PP_data <- beats_indexed[,c(4,1,14)]
head(PP_data)
```

```
## # A tibble: 6 x 3
##      PP   time ann_rel_index
##   <dbl>  <dbl>         <dbl>
## 1  61.0 10001.       NA
## 2  59.3 10001.       NA
## 3  58.5 10002.        0.0961
## 4  58.5 10003.        0.265
## 5  60.0 10004.        0.434
## 6  60.1 10004.        0.603
```

```
PP_gam <- gam(
# The first parameter to the gam() function is the model specification,
# supplied using formula notation:
PP ~ # Left of the tilde (~) is our dependent variable PP
# Right of the tilde is our independent variables.
# Define a smooth function of insp_rel_index.
s(ann_rel_index,
k = 15, # 15 knots.
bs = "cc" # The basis is a cyclic cubic spline
) +
# Define a smooth function of time
s(time,
bs = "cr" # The basis is a natural cubic spline.
# default k is 10. This will be fine here.
),
# We can specify the positions of the knots for each smooth.
# If only two knots are specified for a cyclic spline, these will
# set the positions of the limiting knot(s). The remaining knots will
# be positioned automatically (at quantiles).
knots = list(ann_rel_index = c(0,1)),
# We use restricted maximum likelihood (REML) to fit the optimal smoothing parameter.
# This is often the best choice, but not the default.
method = "REML",
data = PP_data
)
```

```
gratia::draw(PP_gam,
residuals = TRUE)
```



```
coef(PP_gam)[1] # intercept
```

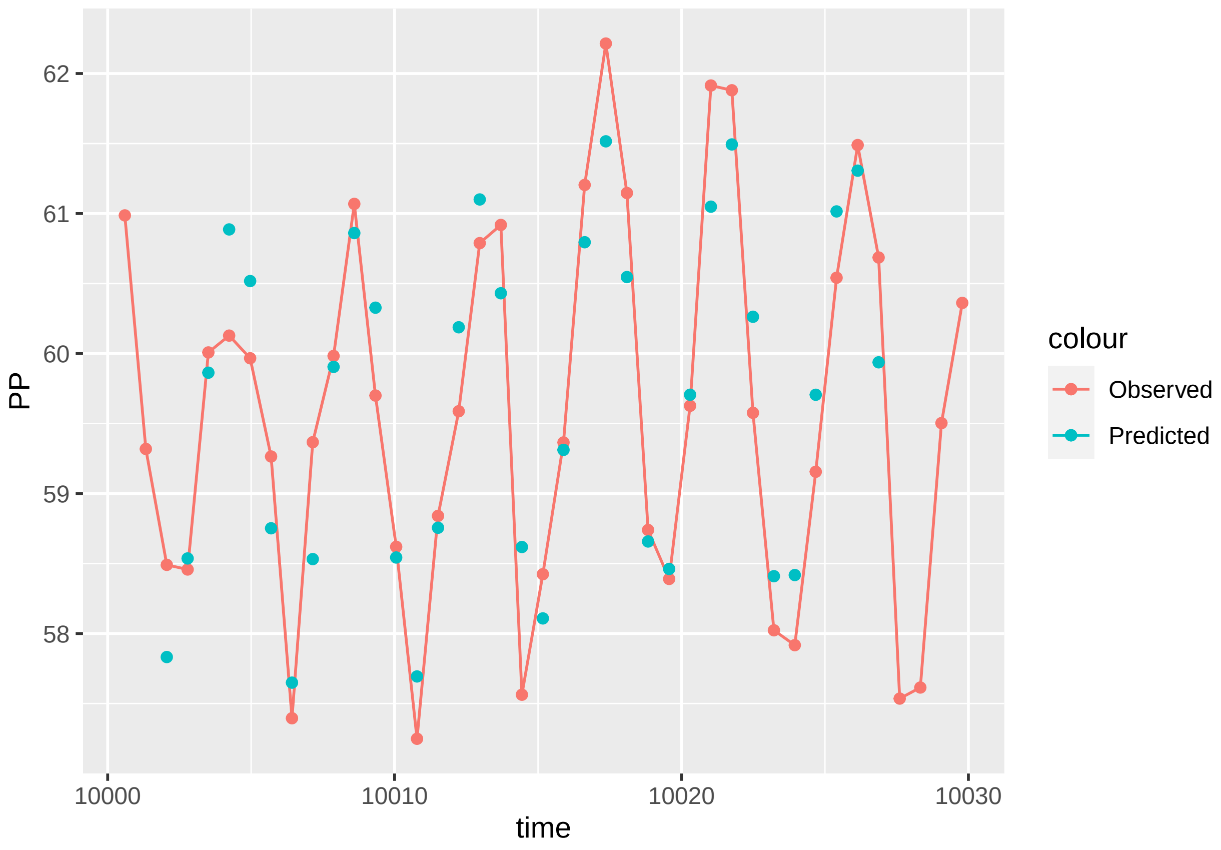
```
## (Intercept)
##    59.64833
```

```
coef(PP_gam)
```

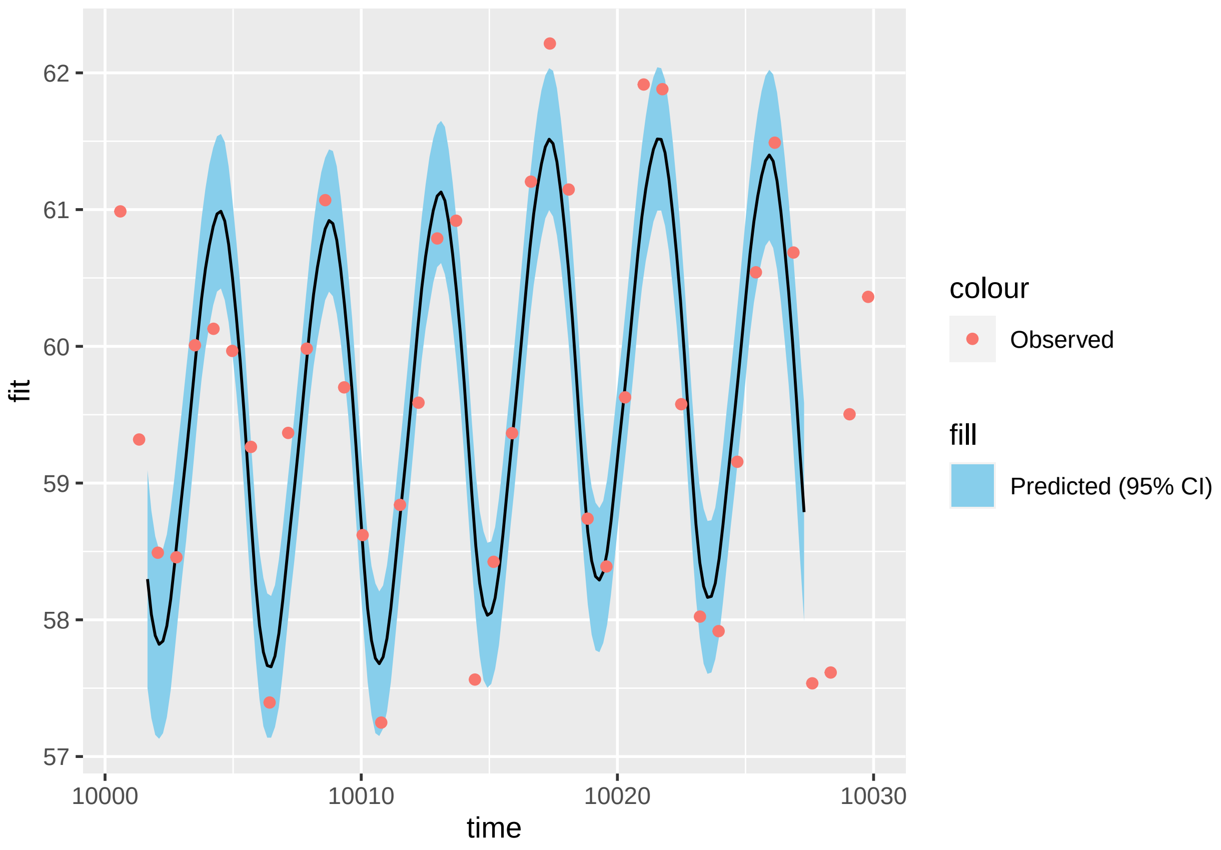
```
##         (Intercept)  s(ann_rel_index).1  s(ann_rel_index).2  s(ann_rel_index).3
##         59.64833166         -1.46657077         -1.35882903         -1.22345446
##  s(ann_rel_index).4  s(ann_rel_index).5  s(ann_rel_index).6  s(ann_rel_index).7
##         -0.44707593         -0.03339568          0.73519444          1.21984685
##  s(ann_rel_index).8  s(ann_rel_index).9 s(ann_rel_index).10 s(ann_rel_index).11
##          1.57750184          1.87789249          1.76245929          1.24615330
## s(ann_rel_index).12 s(ann_rel_index).13           s(time).1           s(time).2
##          0.68156392         -0.45529803         -0.21993067         -0.27994196
##           s(time).3           s(time).4           s(time).5           s(time).6
##         -0.25144792         -0.07663718          0.19474271          0.36494773
##           s(time).7           s(time).8           s(time).9
##          0.33219517          0.22828831          0.17507124
```

```
PP_pred <- mutate(PP_data, pred = predict(PP_gam, PP_data ))
ggplot(PP_pred, aes(x=time)) +
geom_line(aes(y=PP, color = "Observed")) +
geom_point(aes(y=PP, color = "Observed")) +
geom_point(aes(y=pred, color = "Predicted"))
```

```
## Warning: Removed 6 rows containing missing values (geom_point).
```



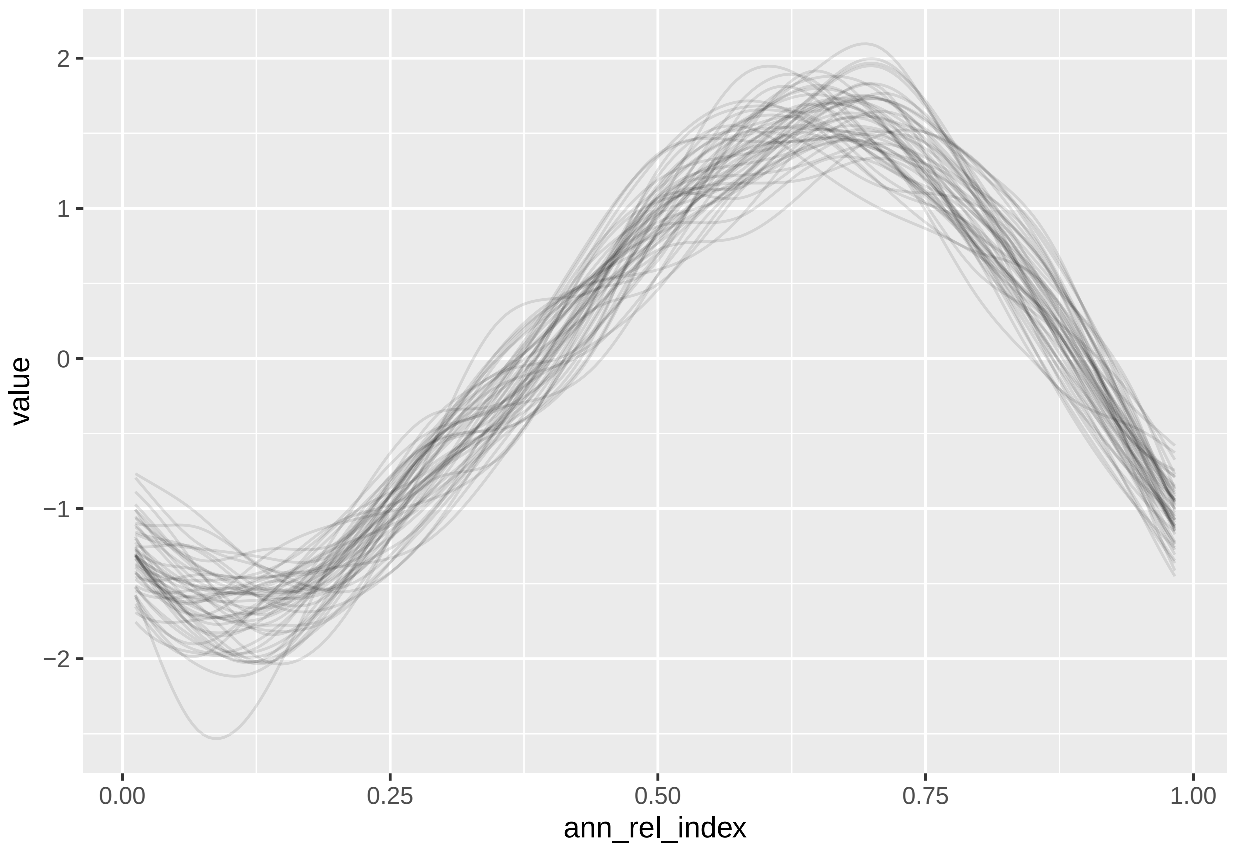
```
PP_newdata <- tibble(
# create 200 points from start to endto get a smooth line
time = seq(start, sec+start, length.out = 200)) %>%
# index each new time to our existing vector of inspiration times
add_time_since_event(insp_start$time, prefix = "ann") %>%
na.omit()
PP_interpolate <- bind_cols(
PP_newdata,
predict(PP_gam,
newdata = PP_newdata,
# in addition to the predictions (fit) also return the standard error
# (se.fit) for each prediction. This makes predict return a named list,
# that we can simply bind to our data frame.
se.fit = TRUE)
)
ggplot(PP_interpolate, aes(x=time)) +
geom_ribbon(aes(ymin = fit - 1.96*se.fit,
ymax = fit + 1.96*se.fit,
fill = "Predicted (95% CI)")) +
geom_line(aes(y = fit)) +
geom_point(aes(y=PP, color = "Observed"), data = PP_data) +
scale_fill_manual(values = "skyblue")
```



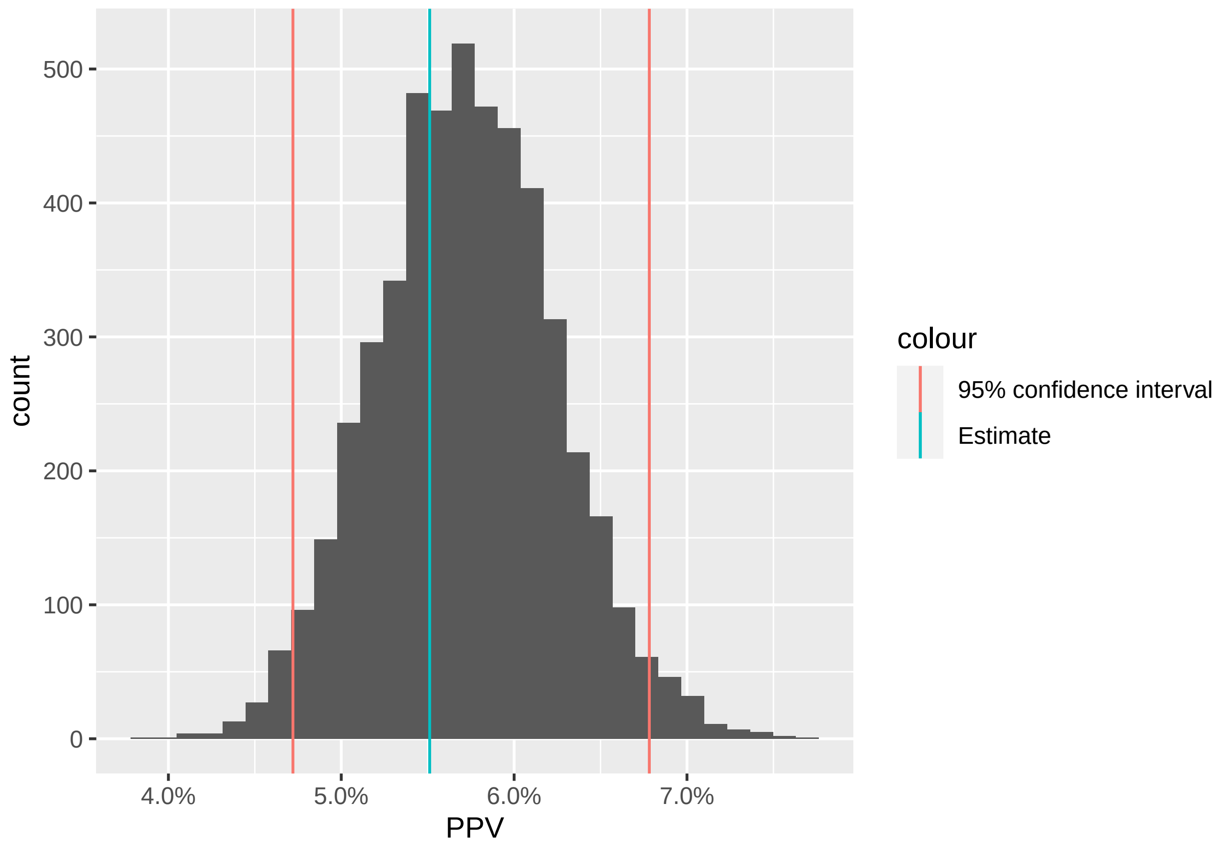
```
insp_rel_index_smooth <- gratia::smooth_estimates(PP_gam,
smooth = "s(ann_rel_index)",
n=100)
min_PP <- min(insp_rel_index_smooth$est)
max_PP <- max(insp_rel_index_smooth$est)
intercept_PP <- coef(PP_gam)[1]
PPV_est <- (max_PP - min_PP) / intercept_PP
sprintf("PPV is %.1f%%", PPV_est*100)
```

```
## [1] "PPV is 5.5%"
```

```
set.seed(1)
insp_smooth_samples <- gratia::smooth_samples(PP_gam, term = "s(ann_rel_index)", n = 5000)
ggplot(insp_smooth_samples %>% filter(draw <= 50), aes(.x1, value, group = draw)) +
geom_line(alpha = 0.1) +
labs(x="ann_rel_index")
```



```r
# A function that returns PPV given a smooth and an intercept
calc_PPV <- function(smooth, intercept) {
  min_PP <- min(smooth)
  max_PP <- max(smooth)
  (max_PP - min_PP) / unname(intercept)
}
PPV_samples <- insp_smooth_samples$value %>%
  split(insp_smooth_samples$draw) %>%
  sapply(calc_PPV, intercept = coef(PP_gam)[1])
  PPV_95 <- quantile(PPV_samples, probs = c(0.025, 0.975))
ggplot(data.frame(PPV = PPV_samples), aes(x=PPV)) +
geom_histogram(bins=30) +
geom_vline(aes(xintercept = PPV_est, color = "Estimate"),
data = data.frame()) +
geom_vline(aes(xintercept = PPV_95, color = "95% confidence interval"),
data = data.frame()) +
scale_x_continuous(labels = scales::label_percent())
```



Udforsk betydningen af outcomes:

Det vi gerne vil nu, er at undersøge betydningen af pulsstrykvariationen på patienters 30 dages mortalitet. Derfor skal vi lave en logistisk regression der ser sådan her ud

$$mortality_i \sim PPV\_thresh_i, art\_mbp_i, Age_i, Sex_i, Asa_i, Eme\_ope_i, BMI_i$$

Derfor skal vi bruge et datasæt der ser sådan her ud:

```r
Case <- c(29,600,1009,1900)
mortality <- c(0,1,1,0)
PPV_thresh <- c(8,9,6,10)
ART_MBP <- c(54,65,69,70)
Age <- c(22,47,87,4)
Sex <- c(1,0,0,1)
Asa <- c(6,4,1,4)
Eme_ope <- c(1,0,0,1) #emergency operation
BMI <- c(32,16,29,15)
data.frame(Case, mortality,PPV_thresh, ART_MBP, Age, Sex, Asa, Eme_ope, BMI)
```

```
##   Case mortality PPV_thresh ART_MBP Age Sex Asa Eme_ope BMI
## 1   29         0          8      54  22   1   6       1  32
## 2  600         1          9      65  47   0   4       0  16
## 3 1009         1          6      69  87   0   1       0  29
## 4 1900         0         10      70   4   1   4       1  15
```

Forklaringer: Case: - Case skal sorteres således at vi kun finder de operationer der er af typen: “open”, “general surgery” og “general Anathesia”. Dette kræver en udvidet load funktion. Der burde være lidt under

3000 observationer

PPV_thresh: - Hvis det så er 180 minutter skal vi fitte 180 gams og udtrække de 180 pulstryk variationer (som er den der amplitude/gennemsnit for pulstryk). Vi vil gerne vide hvor meget af tiden PPV ligger uden for intervallet (6,12) (domain knowledge). Der er lidt forskellige måder dette kan laves på. Men det skal ende ud i enten et samlet tal, eller måske to kolonner med andel af PPV_high, PPV_low. - Vi kan dnytte "anestart" og "aneend" fra clinical information API'en til kun at vælge det interval af operationen hvor patienten er i narkose.

ART_MBP (lav prioritet) : - ART_MBP (mean arterial pressure) er faktisk en del af SOLAR8000, så måske skal vi også begrænse vores cases til det? - Vi er intereserede i hvor ofte ART_MBP ligger under 65 (domain knowledge), så vi skal bare finde andelen af værdierne i ART_MBP under 65

Age, sex, asa, Eme_ops og BMI: - Disse punkter er bare nogle confounders vi skal have med, de ligger i clinical information API'en

Konkrete opgaver (prioriteret): 1. Opgrader load funktion så den kan loade ud fra kriterier i clinical information API - Høj prioritet. Tror jeg (Georg) tager den da jeg har skrevet det meste af load funktionaliteten.

2.Lav en funktion der tager et langt x antal minutter og outputter x PPV-værdier - Høj prioritet

3. Lav funktion, der får et case ID og henter: Age, sex, asa, Eme_ops og BMI

- (Mellem prioritet) Dette kræver ikke at opgave 1 er løst, så den kan en af jer bare tage

4. Udvid subset funktionen så man kan passe at den kun skal kigge på intervallet mellem "anestart" og "aneend". '

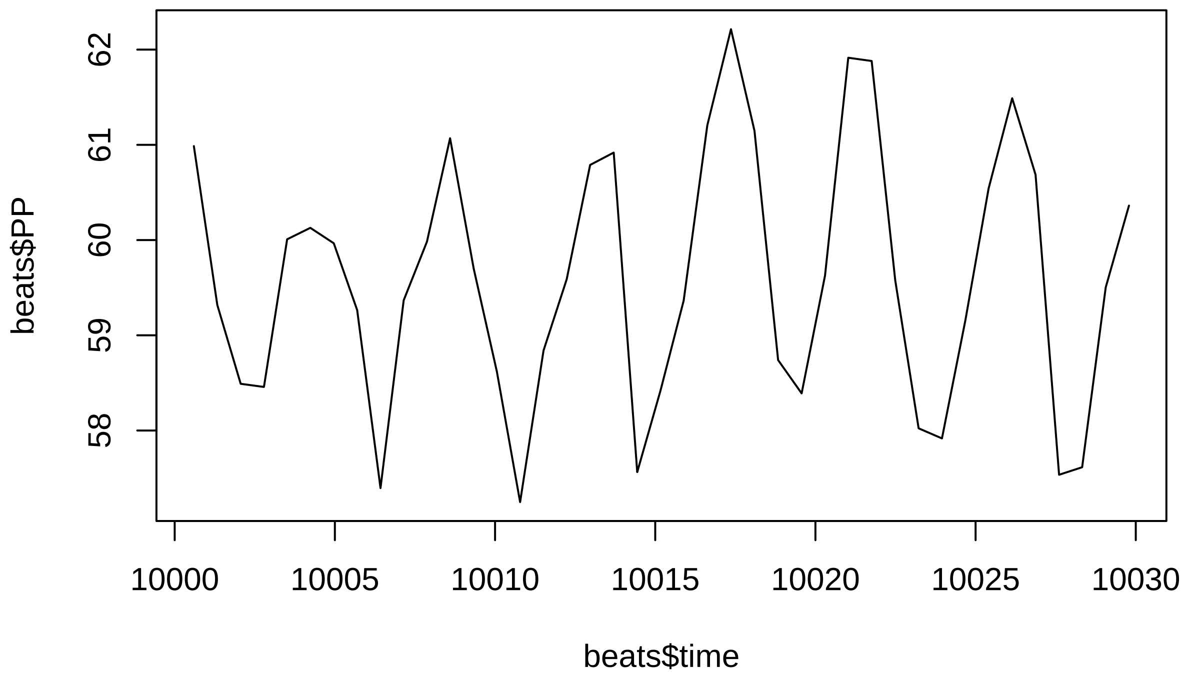
- (lav prioritet) Kræver måske at opgave 1 er løst, men ikke sikkert

Generelt til udvikling: - Fra nu af uploader vi ikke flere funktioner til pakken, men vi holder dem i de dokumenter vi skriver i, så samler vi til sidst, så slipper vi for konstant at skulle pushe til master - Ville være bedst hvis vi stoppede med at pushe til master, men istedet havde hver vores branch vi arbejdede i og så mergede. - Lad os holde vignetten som den er nu, og så ellers lave et dokument for hver funktion man arbejder på, hvor man kopierer de ting man skal bruge fra vignetten ind.

Spørgsmål: - Hvis vi skal bruge ART_MBP skal vi vel også restricte vores cases til dem der er optaget på SOLAR8000? - Skal vi definerer PPV udfra et tal eller to tal som beskrevet i beskrivelsen af PPV_thresh

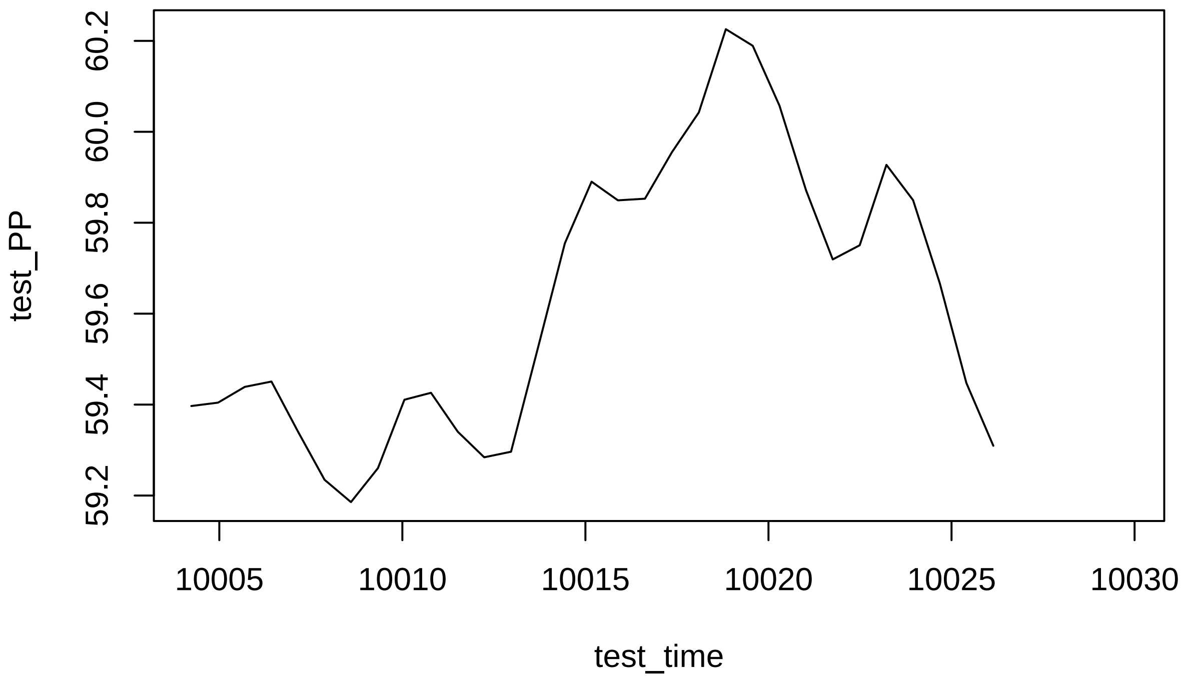
Frederik

```r
# crude plot of Pulse Pressure

plot(beats$time, beats$PP, type='l')
```

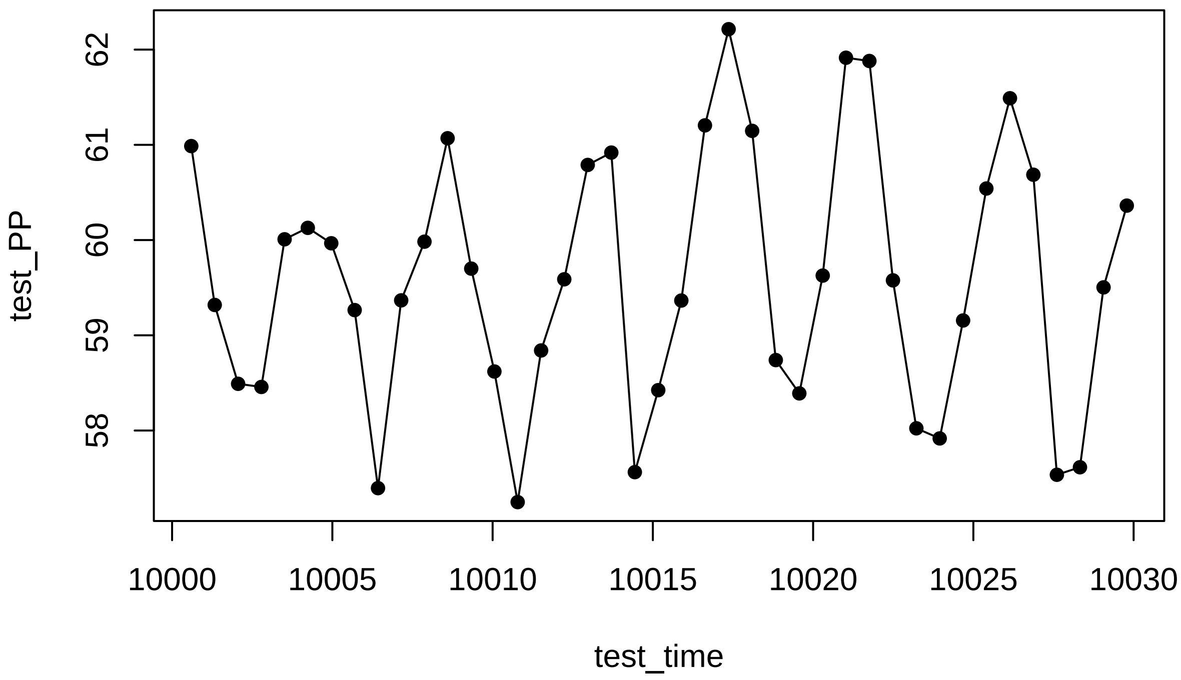


```
#mov_avg <- rep(1/9, 9)
#PP_mov_avg <- filter(beats$PP, mov_avg)
#plot(beats$time, PP_mov_avg, type='l')
```

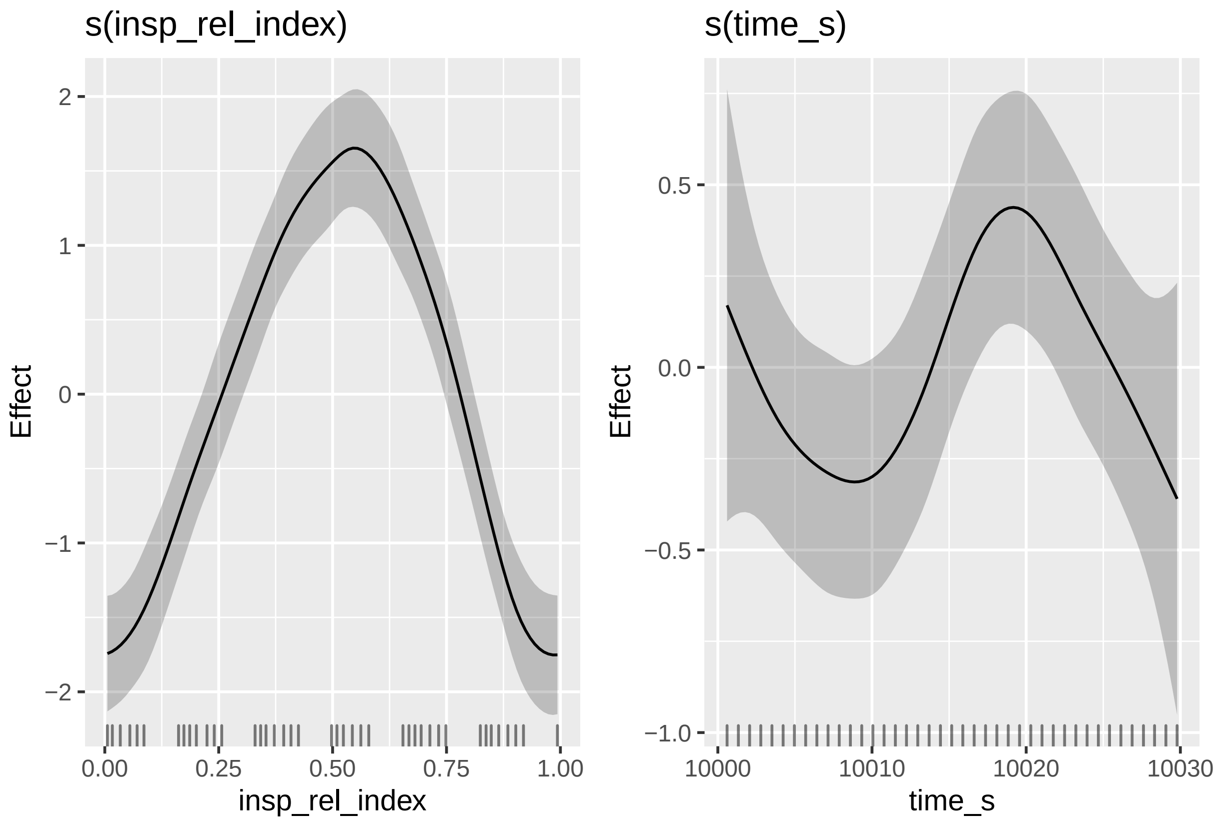
```
# simple average func test (virker ikke bare som sig selv - vi skal have filtreret dataen)

test_time <- beats$time[6:51]
test_PP <- vector(length = 46)
for (i in c(6:51)){
  test_PP[i-5] <- sum(rep(1/11, 11) * beats$PP[(i-5):(i+5)])
}

plot(test_time, test_PP, type='l')
```



```r
# cheating (fjern alle værdier <15, da det virker til at være den slags støj vi finder mest her)
test_time <- beats$time
test_PP <- beats$PP

while (test_PP[1] < 15) {
  test_time <- test_time[2:length(test_time)]
  test_PP <- test_PP[2:length(test_PP)]
}

while (min(test_PP, na.rm = TRUE) < 15) {
  idx <- which.min(test_PP)
  test_time <- c(test_time[1:(idx-1)], test_time[(idx+1):length(test_time)])
  test_PP <- c(test_PP[1:(idx-1)], test_PP[(idx+1):length(test_PP)])
}

plot(test_time, test_PP, type='l')
points(test_time, test_PP, pch = 16, col='black')
```



```
#abline(v = insp_start, col='red')

#GAM plot uden inspiration
waveformtools::PP_gam_analysis(beats, return_GAM = TRUE)|>  gratia::draw()
```



```
# FRED COMM: Det virker til at der er lidt delay på PP / ins_start kommer for tidligt.
# Vi ville ellers "gerne se" at ins_start kommer samtidig med stigning i PP, men den kommer mest før
```