Exam: First Session

Nonparametric Statistics, AY 2020/21

January 22, 2021

Algorithmic Instructions

- All the numerical values required need to be put on an A4 sheet and uploaded, alongside the required plots.
- For all computations based on permutation/resampling, as well as split conformal, use $B = 200$ replicates, and $seed = 100$.
- Both for confidence and prediction intervals, as well as tests, set $\alpha = 0.05$.
- When reporting a test result, please specify H_0, H_1, the $P-$value and the corresponding conclusion.
- When reporting confidence/prediction intervals, always provide upper and lower bounds.

Exercise 1

Dr. Simoni, a data scientist, just had his second kid (a boy, 58 cm tall) and he is interested to know how tall his child will be at 25 years of age, given his height at birth. To do so, he has collected the height at birth of 100 males, alongside the height they reached when they were 25. He has collected these data in the file `boyheight.rda`, `height.25` is the height [cm] at 25 years of age, while `height.b` is the length of the newborn [cm]. Assume that $height25 = f(height.b) + \varepsilon$. Now:

1. Build a degree 1 regression spline model, with breaks at the 25th and 75th percentile of `height.b`, to predict the height at 25 from the height at birth. Provide a plot of the regression line, compute the pointwise prediction (round it at the second decimal digit) for the height at 25 of Dr. Simoni newborn child, and calculate, using a bootstrap approach on the residuals, the bias, variance and Mean Squared Error (MSE) of such prediction.
2. Dr. Simoni is not particularly satisfied with the predictions of such a simple model, and would like you to do more. So build a prediction model for the height at 25 based on a smoothing spline of order 4 (select the lambda parameter via Leave-One-Out CV). Report the optimal lambda value (2 decimal digits), provide a plot of the regression line, alongside the pointwise prediction of the height at 25 of Dr. Simoni's kid, and

calculate using a bootstrap approach on the residuals the bias, variance and MSE of such prediction (fix the lambda value to the one obtained via Leave-One-Out CV).

3. Like every good statistician, Dr. Simoni is not satisfied with just a pointwise prediction: After having stated the distributional hypothesis behind this approach, build split conformal prediction intervals for the the height at 25 of Dr. Simoni's kid, using the best model (selected using the MSE), the absolute value of the regression residuals as the non-conformity score and an equal split.

Exercise 2

In the file `clients.rda` you can find data about the churning of customers of a streaming television service. After the amount of days indicated by the `time` variable some of the customers have cancelled their subscription (if `status= 2`), while some others are still active (if `status= 1`). Alongside this kind of information, you are also given data about the income and age of the subscriber, and a grouping based on behavioural segmentation of your customers.

1. Assuming all `(age, income)` tuples to be independent, and covariate data within behavioural groups also identically distributed, check, via a permutation test using as a test statistic the maximum norm of the absolute difference between the sample multivariate Mahalanobis medians of the groups, if the two groups, that are different in terms of behaviour, also have a different median for both age and income. Plot the empirical cumulative distribution function of the permutational test statistic, report the p-value of the test and comment it.

2. Compute for the two behavioural groups the Kaplan-Meier estimation of the survival curve, and plot the two curves. Test also if the time-to-event distributions of the two behavioural groups are equal via a Log-rank test, and comment the result.

3. Provide a confidence interval, using the Tsiatis formula (i.e. the default setting in R) for each behavioural segment[1], in correspondence of the median unsubscription time, for a person whose age and income are equal to the multivariate Mahalanobis median of the segments (but do not forget what you tested at point 1 of this exercise).

[1] Remember that you can estimate a Cox proportional hazard model in the R programming language with `model=coxph(formula,data=data)` and you can generate prediction intervals from this model with `survfit(model,newdata=newdata)`, where the `newdata` argument needs to be a dataframe with the same labels used in the coxph formula

First Session - Solution

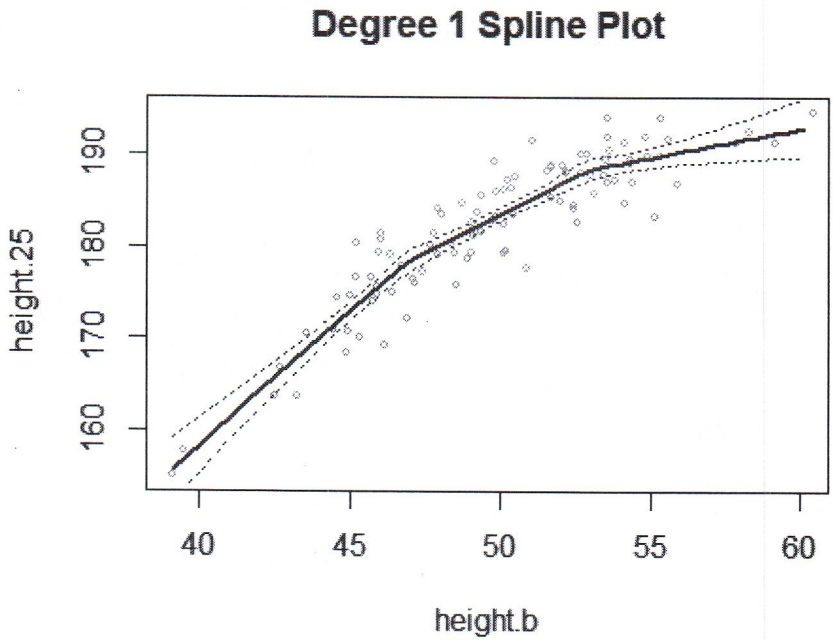
Matteo Fontana

22/01/2021

Exercise 1

Point 1

Plot of the degree 1 regression spline with knots at 25^{th} and 75^{th} percentile.



Pointwise prediction for = 58, rounded at second decimal digit

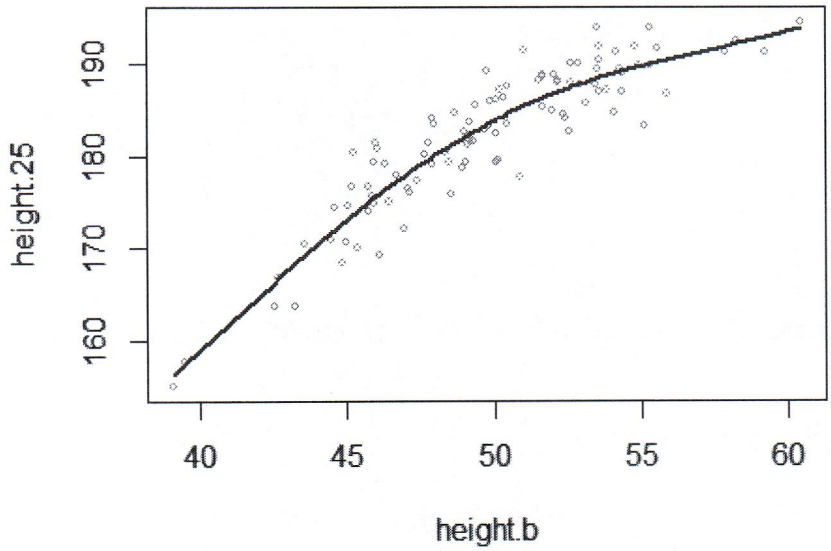
```
##      1
## 191.43
```

Bias, Variance and MSE via residuals bootstrapping

```
##          bias variance     MSE
## 1 -0.04960756 1.158379 1.16084
```

Point 2

Plot of order 4 smoothing spline, optimal lambda selected via LOOCV



Optimal lambda and pointwise prediction

```
## Optimal Lambda
##           0.01

## Pointwise Prediction
##              191.97
```

Bias, Variance, MSE via Residual Bootstrap:

```
##        bias  variance      MSE
## 1 0.3520232 0.9159435 1.039864
```

Point 3

Choose the best model:

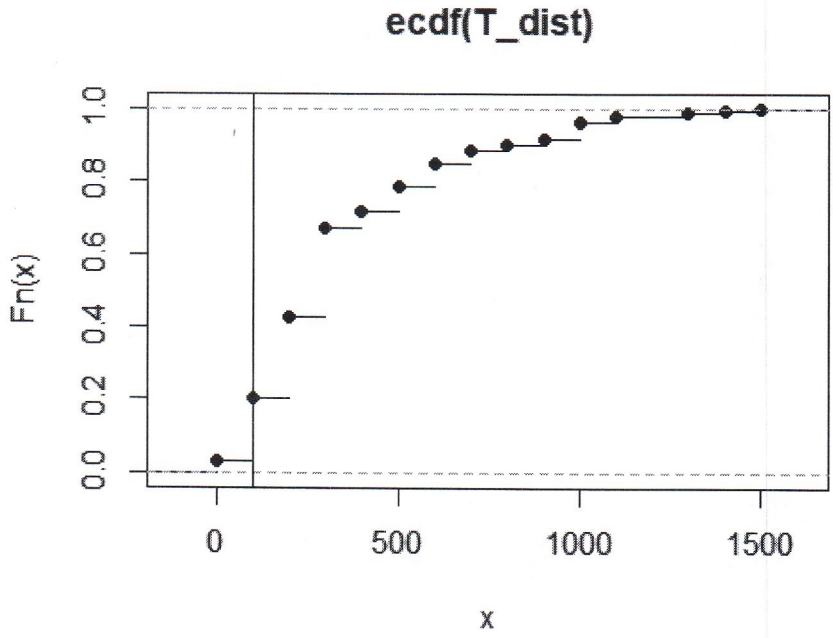
```
##   MSE_Smooth MSE_Deg1
## 1   1.039864  1.16084
```

The hypothesis needed here is that $(height.b, height.25)_i$ are iid.

The best model is the smooth one, generate predictions from it:

```
##       lwr     pred      upr
## 1 185.929 192.7354 199.5418
```

##Exercise 2 ##Point 1 Permutation test: H_0: $Me(group1) = Me(group2)$ vs H_0: $Me(group1) \neq Me(group2)$ where Me is the multivariate Mahalanobis median



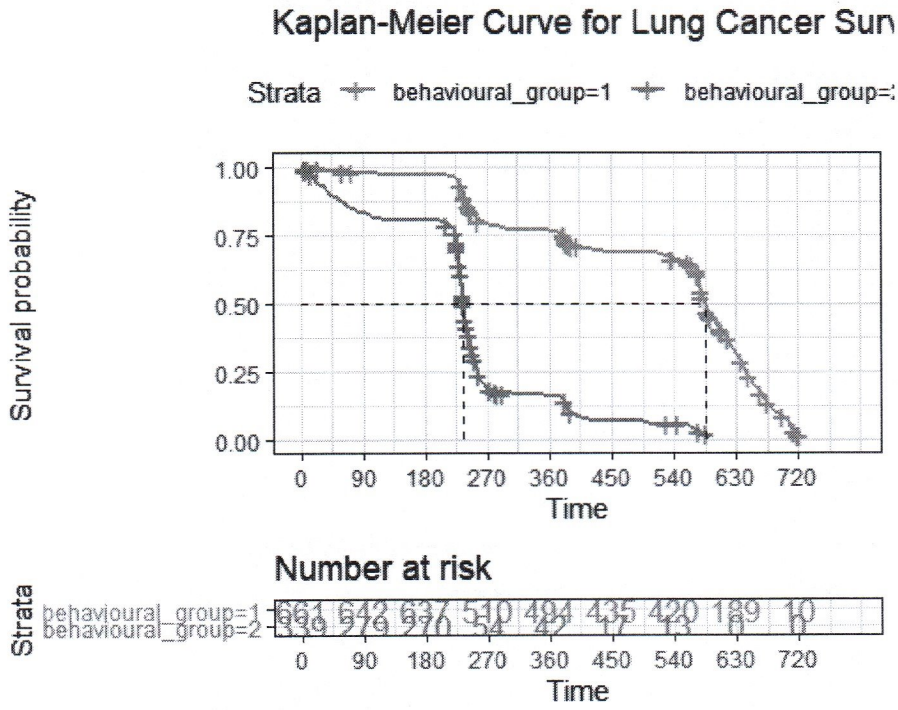
The P −value of the test is:

```
## [1] 0.97
```

The P −value is $>> 0.05$, I cannot refuse H_0, which means that I cannot refuse the equality of the two Mahalanobis medians.

##Point 2

Survival Curves for the two behavioural groups computed using the Kaplan-Meier estimator



log-rank test between the two behavioural groups: $H_0: S_1(t) = S_2(t), H1: S_1(t) \neq S_2(t)$

```
## Call:
## survdiff(formula = Surv(time, status) ~ behavioural_group, data = dt_2)
##
##                          N Observed Expected (O-E)^2/E (O-E)^2/V
## behavioural_group=1 661      609      813        51       542
## behavioural_group=2 339      304      100       412       542
##
##  Chisq= 542  on 1 degrees of freedom, p= <2e-16
```

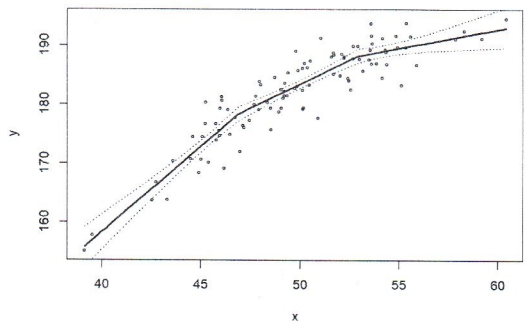
The P-value is a numerical 0, so we reject H_0, the two survival distributions are statistically different. The two behavioural groups have a very different churn behaviour, while being equal in terms of median age and income.

##Point3

The median of age and income is the same for the two behavioural groups, so I compute only one median, without group splitting:

```
## Call: survfit(formula = cox.age, newdata = newdata)
##
##       n events median 0.95LCL 0.95UCL
## 1 1000    913    587     582     592
## 2 1000    913    236     234     238
```

```r
#-------------------------------------------------------------------------------
# Exercise 1
#-------------------------------------------------------------------------------
load('boyheight.rda')
# y = f(x) + eps
y = height.25
x = height.b

seed  = 100
B     = 200
alpha = 0.05

#----#
# 1. #
#----#
library(splines)
degree   = 1
br       = c(quantile(x, probs=c(0.25, 0.75)))
model_sp = lm(y ~ bs(x, degree=degree, knots=br))
x_grid   = seq(range(x)[1], range(x)[2], length.out=100)
y_pred   = predict(model_sp, list(x=x_grid), se=T)
se.bands = cbind(y_pred$fit+2*y_pred$se.fit, y_pred$fit-2*y_pred$se.fit)
plot(x, y, xlim=range(x_grid), cex=.5)
lines(x_grid, y_pred$fit, lwd=2, col='blue')
matlines(x_grid, se.bands, lwd=1, col='blue', lty=3)
```



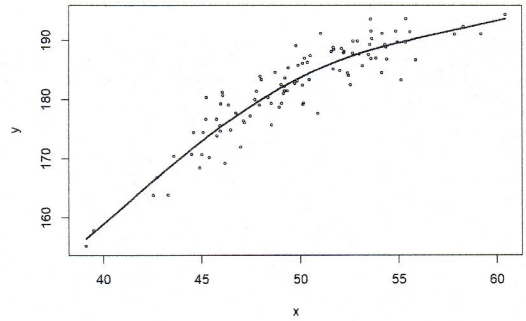

```r
# Prediction
x_new = 58
y_new = predict(model_sp, list(x=x_new))
y_new
```

```
##        1
## 191.4279
```

```r
# Bootstrap
fm         = lm(y ~ bs(x, degree=degree, knots=br))
T0         = predict(fm, list(x=x_new))
T_boot     = numeric(B)
fitted_obs = fitted(fm)
res_obs    = residuals(fm)

set.seed(seed)
for(k in 1:B){
  y_boot    = fitted_obs + sample(res_obs, replace=T)
  fm_boot   = lm(y_boot ~ bs(x, degree=degree, knots=br))
  T_boot[k] = predict(fm_boot, list(x=x_new))
}

# Bootstrap var, bias, MSE
VAR_1  = var(T_boot)
BIAS_1 = mean(T_boot) - T0
MSE_1  = sd(T_boot)^2 + (mean(T_boot)-T0)^2

data.frame(cbind(VAR_1, BIAS_1, MSE_1))
```

```
##      VAR_1      BIAS_1   MSE_1
## 1 1.158379 -0.04960756 1.16084
```

```r
#----#
# 2. #
#----#
# Fix the lambda
fit    = smooth.spline(x, y, cv=T)  # CV
lambda = fit$lambda
fit    = smooth.spline(x, y, lambda = lambda)
plot(x, y, cex=.5)
lines(fit, col='blue', lwd=2)
```




```
# Prediction
val = 58
predict(fit, val)$y
```

```
## [1] 191.9669
```

```
# Bootstrap
T0          = predict(fit, val)$y
T_boot      = numeric(B)
fitted_obs = fitted(fit)
res_obs     = residuals(fit)

set.seed(seed)
for(k in 1:B){
  y_boot    = fitted_obs + sample(res_obs, replace=T)
  fm_boot   = smooth.spline(x, y_boot, lambda=lambda)
  T_boot[k] = predict(fm_boot, x=val)$y
}

# Bootstrap var, bias, MSE
VAR_2  = var(T_boot)
BIAS_2 = mean(T_boot) - T0
MSE_2  = sd(T_boot)^2 + (mean(T_boot)-T0)^2

data.frame(cbind(VAR_2, BIAS_2, MSE_2))
```

```
##       VAR_2    BIAS_2    MSE_2
## 1 0.9159435 0.3520232 1.039864
```

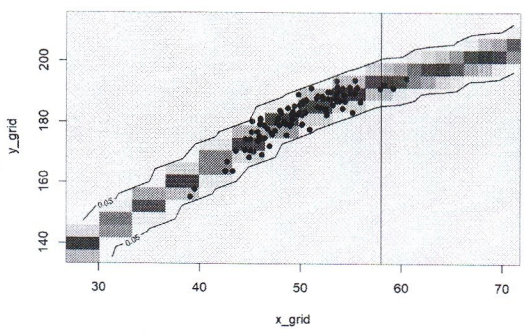
```
#----#
# 3. #
#----#
data.frame(cbind(VAR_1, BIAS_1, MSE_1))
```

```
##      VAR_1      BIAS_1   MSE_1
## 1 1.158379 -0.04960756 1.16084
```

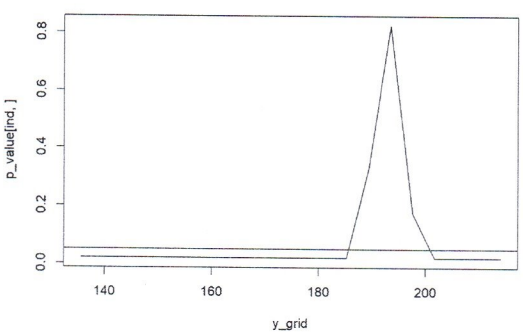
```
data.frame(cbind(VAR_2, BIAS_2, MSE_2))
```

```
##       VAR_2    BIAS_2    MSE_2
## 1 0.9159435 0.3520232 1.039864
```

```
# We choose the second model

y          = height.25
x          = height.b
data       = as.data.frame(cbind(x,y))
n          = dim(data)[1]
train_prop = 0.5

set.seed(seed)
train_id   = sample(1:n, ceiling(n*train_prop), replace=F)
train_set  = data[train_id,]
calib_set  = data[-train_id,]
x          = calib_set[,1]
y          = calib_set[,2]
x_grid     = c(seq(min(x)-0.5*diff(range(x)), 58, length.out=10),
               seq(58+diff(range(x))/50, max(x)+0.5*diff(range(x)), length.out=10))
y_grid     = seq(min(y)-0.5*diff(range(y)), max(y)+0.5*diff(range(y)), length.out=20)
p_value    = matrix(nrow=length(x_grid), ncol=length(y_grid))

# Model on training
fm = smooth.spline(train_set[,1], train_set[,2], lambda = lambda)

# Conformal Intervals
NC = function(z_aug, i){
  abs(z_aug[i,2] - predict(fm, x = z_aug[i,1])$y)
}

for(k in 1:length(x_grid)){
  for(h in 1:length(y_grid)){
    data_aug = rbind(calib_set, c(x_grid[k], y_grid[h]))
    scores   = numeric(dim(data_aug)[1])
    for(i in 1:length(scores)){
      scores[i] = NC(data_aug, i)
    }
    p_value[k,h] = sum(scores>=scores[dim(data_aug)[1]])/(dim(data_aug)[1])
  }
}

# Plot of the p-value
image(x_grid, y_grid, p_value, zlim=c(0,1))
points(data, pch=16)
contour(x_grid, y_grid, p_value, levels=alpha, add=T)
abline(v=58)
```




```
# Focus on value 58
ind = which(x_grid==58)
plot(y_grid, p_value[ind,], type='l')
abline(h=alpha, col='red')
```




```
PI_grid = y_grid[which(p_value[ind,]>=alpha)]
PI      = c(min(PI_grid), max(PI_grid))
PI
```

```
## [1] 189.2790 197.5407
```

```r
#-------------------------------------------------------------------------------------------
# Exercise 2
#-------------------------------------------------------------------------------------------
load('clients.rda')
data = dt_2
head(data)
```

```
##   age income behavioural_group time status
## 1  45  30200                 2    8      2
## 2  34  28500                 1   13      2
## 3  26  49400                 1  603      2
## 4  40  33900                 1  666      1
## 5  40  41000                 1  387      2
## 6  34  41800                 1  385      2
```

```r
data$behavioural_group = as.factor(data$behavioural_group)

data_1 = data[which(data$behavioural_group==1),c(1,2)]
data_2 = data[which(data$behavioural_group==2),c(1,2)]

#----#
# 1. #
#----#
library(DepthProc)
```

```r
method_name = 'Mahalanobis'

seed = 100
B    = 200

n1        = dim(data_1)[1]
n2        = dim(data_2)[1]
n         = n1+n2
data_pool = rbind(data_1, data_2)

# CMC to estimate the p-value
set.seed(seed)
data_1_med = depthMedian(data_1, depth_params = list(method=method_name))
data_2_med = depthMedian(data_2, depth_params = list(method=method_name))
T0         = max(abs(data_1_med - data_2_med))
T_stat     = numeric(B)

for(k in 1:B){
  perm            = sample(1:n)
  data_pool_perm  = data_pool[perm,]
  data_1_perm     = data_pool_perm[1:n1,]
  data_2_perm     = data_pool_perm[(n1+1):n,]
  data_1_perm_med = depthMedian(data_1_perm, depth_params = list(method=method_name))
  data_2_perm_med = depthMedian(data_2_perm, depth_params = list(method=method_name))
  T_stat[k]       = max(abs(data_1_perm_med - data_2_perm_med))
}

# Permutational distribution of T
plot(ecdf(T_stat), xlim=range(c(T_stat,T0)))
abline(v=T0, col=3, lwd=2)

# p-value
p_value = sum(T_stat>=T0)/B
p_value
```

```
## [1] 0.975
```

```r
#----#
# 2. #
#----#
library(survival)
```

```r
library(survminer)
```

```r
library(dplyr)
```

```r
# Kaplan-Meier estimator for survival curve
feature     = data$behavioural_group
fit.feature = survfit(Surv(time, status) ~ feature, data=data)
print(fit.feature)
```

```
## Call: survfit(formula = Surv(time, status) ~ feature, data = data)
##
##              n events median 0.95LCL 0.95UCL
## feature=1 661    609    587     581     592
## feature=2 339    304    236     234     239
```

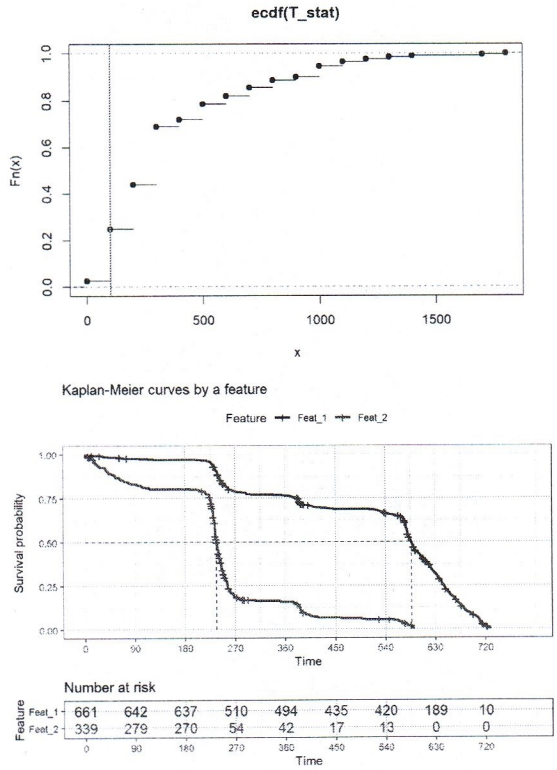
```r
summary(fit.feature)$table
```

```
##           records n.max n.start events   *rmean *se(rmean) median 0.95LCL
## feature=1     661   661     661    609 513.5711   7.096029    587     581
## feature=2     339   339     339    304 238.2945   7.315692    236     234
##           0.95UCL
## feature=1     592
## feature=2     239
```

```
summary(fit.feature)
```

```
## Call: survfit(formula = Surv(time, status) ~ feature, data = data)
##
##                feature=1
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##     4    661       1  0.99849 0.00151     0.995529       1.0000
##     6    660       1  0.99697 0.00214     0.992796       1.0000
##    13    658       1  0.99546 0.00262     0.990346       1.0000
##    ..
```

```
# Kaplan-Meier curves plot by a feature
ggsurvplot(fit.feature, data = data,
           conf.int = F,
           risk.table = TRUE,
           risk.table.col = "strata",
           surv.median.line = "hv",
           ggtheme = theme_bw(),
           break.time.by = 90,
           legend.labs = c("Feat_1", "Feat_2"),
           legend.title = "Feature",
           palette = c("blue", "red"),
           title = 'Kaplan-Meier curves by a feature')
```




```
# Log-rank test for a feature
log_rank = survdiff(Surv(time, status) ~ feature, data=data)
log_rank
```

```
## Call:
## survdiff(formula = Surv(time, status) ~ feature, data = data)
##
##              N Observed Expected (O-E)^2/E (O-E)^2/V
## feature=1 661      609      813        51       542
## feature=2 339      304      100       412       542
##
##  Chisq= 542  on 1 degrees of freedom, p= <2e-16
```

```
#----#
# 3. #
#----#
cox_mult = coxph(Surv(time, status) ~ age + income + behavioural_group, data=data)
data_med = depthMedian(data[,c(1,2)], depth_params = list(method=method_name))

new_data = rbind(data_med, data_med)
new_data = as.data.frame(new_data)
new_data$behavioural_group = as.factor(c(1,2))

survfit(cox_mult, newdata=new_data)
```

```
## Call: survfit(formula = cox_mult, newdata = new_data)
##
##                n events median 0.95LCL 0.95UCL
## data_med    1000    913    587     582     592
## data_med.1 1000    913    236     234     238
```