R Notebook

This is an R Markdown Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Ctrl+Shift+Enter*.

```
library(survival)
# library(condSURV)
# library(JM)
# library(dplyr)
# library(survminer)
# library(clustcurv)
library(psych)
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following objects are masked from 'package:psych':
##
##     %+%, alpha
```

This downloads the data, and edits it a bit. In partifcular loan status is transformed, variables are turned into dates, the year 2006 is selected etc. This is from an online course on survival data.

```
web <- "https://s3.amazonaws.com/udacity-hosted-downloads/ud651/prosperLoanData.csv"
loan <- read.csv(web)

# Remove duplicates by LoanKey
loan_nd <- loan[!duplicated(loan$LoanKey), ]

# removing LoanStatus no needed
sel_status  <- loan_nd$LoanStatus %in% c("Completed", "Current",
                                         "ChargedOff", "Defaulted",
                                         "Cancelled")
loan_filtered <- loan_nd[sel_status, ]

# creating status variable for censoring
loan_filtered$status <- ifelse(
  loan_filtered$LoanStatus == "Defaulted" |
    loan_filtered$LoanStatus == "Chargedoff",  1, 0)

# adding the final date to "current" status
head(levels(loan_filtered$ClosedDate))
```

```
## NULL
```

```
## [1] ""                   "2005-11-25 00:00:00" "2005-11-29 00:00:00"
## [4] "2005-11-30 00:00:00" "2005-12-08 00:00:00" "2005-12-28 00:00:00"
levels(loan_filtered$ClosedDate)[1] <- "2014-11-03 00:00:00"
```

```
# creating the time-to-event variable
loan_filtered$start <- as.Date(loan_filtered$LoanOriginationDate)
loan_filtered$end <- as.Date(loan_filtered$ClosedDate)
loan_filtered$time <- as.numeric(difftime(loan_filtered$end, loan_filtered$start, units = "days"))

# there is an error in the data (time to event less than 0)
loan_filtered <- loan_filtered[-loan_filtered$time < 0, ]

# just considering a year of loans creation
ii <- format(as.Date(loan_filtered$LoanOriginationDate),'%Y') %in% c("2006")
loan_filtered <- loan_filtered[ii, ]

loan_filtered$LoanOriginalAmount2 <-  loan_filtered$LoanOriginalAmount/10000


# write.csv(to_save_data, 'loan_data_2006_cleaned')
```

```
# describe(loan_filtered)
```

I did not really analyse the entire dataset much. You'd have to filter it quite carefully. Many NA's are present. Also many ' ' empty strings are present.

```
# print(sum(!complete.cases(loan_data_2006[,"BorrowerState"])))
```

```
# # print(colSums(is.na(loan_data_2006)))
# loan_data_2006 = loan_data_2006_cleaned[, colSums(is.na(loan_data_2006_cleaned)) == 0]
# # loan_data_2006 = loan_data_2006[, colSums(is.na(loan_data_2006)) == 0]
# print(colSums(loan_data_2006 == '') == 0)
```

Select the subset of the data:

```
subset_data = loan_filtered[, c("IsBorrowerHomeowner", "LoanOriginalAmount2", 'time', 'status')]
```

Fit a cox model without including spline terms on all data (4954 loans, of which 1373 are observed events. I.e. 28 % of the events is observed.). This results in significant effect. Overall p-value is p=4.358e-06.

```
mfit <- coxph(Surv(time, status) ~ IsBorrowerHomeowner + LoanOriginalAmount2, data=subset_data)
mfit
```

```
## Call:
## coxph(formula = Surv(time, status) ~ IsBorrowerHomeowner + LoanOriginalAmount2,
##     data = subset_data)
##
##                              coef exp(coef) se(coef)      z        p
## IsBorrowerHomeownerTrue -0.24926   0.77938  0.06210 -4.014 5.97e-05
## LoanOriginalAmount2     -0.13293   0.87553  0.06649 -1.999   0.0456
##
## Likelihood ratio test=24.69  on 2 df, p=4.358e-06
## n= 4954, number of events= 1373
```

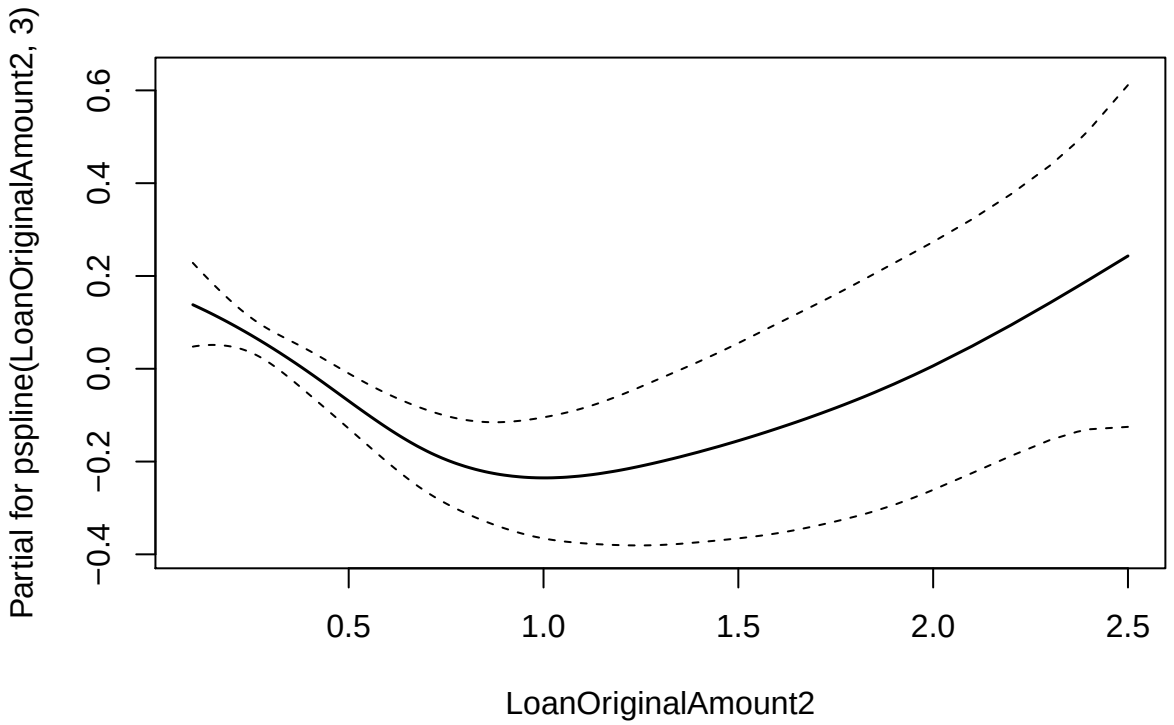
Now we fit again a model on all the data, but we include spline terms for the LoanOriginalAmount2. With a spline of order 3 the overall p-value is p=9e-09. The plot seems to suggest medium sized loans have the least risk to default.

```
mfit <- coxph(Surv(time, status) ~ IsBorrowerHomeowner + pspline(LoanOriginalAmount2, 3) , data=subset_
mfit
```

```
## Call:
```

```
## coxph(formula = Surv(time, status) ~ IsBorrowerHomeowner + pspline(LoanOriginalAmount2,
##     3), data = subset_data)
##
##                               coef se(coef)     se2   Chisq   DF       p
## IsBorrowerHomeownerTrue   -0.2378   0.0623  0.0623 14.5902 1.00 0.00013
## pspline(LoanOriginalAmoun -0.1072   0.0614  0.0609  3.0474 1.00 0.08087
## pspline(LoanOriginalAmoun                           17.5656 2.09 0.00017
##
## Iterations: 6 outer, 14 Newton-Raphson
##      Theta= 0.979
## Degrees of freedom for terms= 1.0 3.1
## Likelihood ratio test=43.5  on 4.09 df, p=9e-09
## n= 4954, number of events= 1373
```

```
termplot(mfit, term=2, se=TRUE, col.term=1, col.se=1)
```




```
ptemp <- termplot(mfit, se=TRUE, plot=FALSE)
```

We now plot the two KM curves for the binary variable IsBorrowerHomeowner. Actually the CPH assumption seems quite fine, except for a couple of observations at the end.

```
km_trt_fit <- survfit(Surv(time, status) ~ IsBorrowerHomeowner, data=subset_data)
# autoplot(km_trt_fit)
```

Now we do the analysis for smaller sample sizes:

```
mfit <- coxph(Surv(time, status) ~ IsBorrowerHomeowner + LoanOriginalAmount2 , data=subset_data[sample(
mfit
```

```
## Call:
## coxph(formula = Surv(time, status) ~ IsBorrowerHomeowner + LoanOriginalAmount2,
##     data = subset_data[sample(nrow(subset_data), 1000), ])
##
##                             coef exp(coef) se(coef)      z     p
```

```
## IsBorrowerHomeownerTrue -0.1548    0.8566   0.1293 -1.197 0.231
## LoanOriginalAmount2     -0.2295    0.7949   0.1632 -1.406 0.160
##
## Likelihood ratio test=4.53  on 2 df, p=0.1037
## n= 1000, number of events= 301
```

And finally we average the p-value obtained on the subsamples over 100 runs. In Python I get {'cph_test': 0.13403668092069104, 'gaucon': 0.04515484515484516, 'gaugau': 0.0073426573426573424}

```
p_value_sum = 0
for (i in 0:100){
  mfit <- coxph(Surv(time, status) ~ IsBorrowerHomeowner + LoanOriginalAmount2 , data=subset_data[sampl
  p_value = summary(mfit)$sctest['pvalue']
  p_value_sum = p_value_sum + p_value
}
print(p_value_sum/100)
```

```
##    pvalue
## 0.1292787
```

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Ctrl+Shift+K* to preview the HTML file).

The preview shows you a rendered HTML copy of the contents of the editor. Consequently, unlike *Knit*, *Preview* does not run any R code chunks. Instead, the output of the chunk when it was last run in the editor is displayed.