# Cox model Assumptions

## Proportional

That the stratified variables will increase proportionally. The shape of the function over time does not matter. For example with male and females the difference in hazard at time t1 will be the same as at time t2 and t3­­.

## Linearity

Cox model assumes that for each predictor there is one parameter. Therefore when plotting residuals this should follow the predicted coefficient. If this does not occur then we assume that the predictor is nonlinear to the base line hazard.

**Relaxation of linearity**

It is possible to relax the linearity assumption using **restricted cubic splines** as shown in Harrell case study using prostate cancer data, where certain predictors are incorporated into the model despite that data not interacting with the base line hazard in a linear manner

Therefore if you find that you have a non-linear parameter it is possible to correct for this using the restricted cubic spline function **rcs()**. Deciding how many knots to include in your restricted cublic spline should not need more than 4 knots as this can capture variability, but need to be careful not to overfit. (Number of knots required can be gauged latter when looking at anova results).

# Determining the number of parameters/degrees of freedom in your model without over fitting

To prevent overfitting the data, the book suggests that generally the maximum number of predictors that can be used is number of events which took place divide by 15 to 20. (depending on stringency). This number is a empirical value but has been commonly accepted as a good rule of thumb.

So in the given example there was a population of 502 but 354 patients died in the given time frame of the study. Therefore the number of predictors available are

354/15 ~= 23

Or more stringently

354/20 = 18

# Case Study

Extract from the Frank Harrell’s Regression Modelling Strategies Textbook, where they were looking at survival of prostate cancer. Which included numerous predictors:

|  |  |  |  |
| --- | --- | --- | --- |
| patno | Patient number | datatype | d.f |
| stage | Cancer stage | numerical | 1 |
| rx | Treatment – dose of estrogen | categorical | 4 |
| dtime | Time of death or censoring (months) | Numerical |  |
| Status | Event occurred? | Logical | 1 |
| Age | Age | numerical |  |
| wt | Wt index | Numerical |  |
| pf | Performance rating | Categorical | 2 |
| Hx | History of cardiovascular | Logical | 1 |
| Sbp | Systolic blood pressure | numerical |  |
| Dbp | Dystolic blood pressure | Numerical |  |
| Exg | electrocardiogram | Categorical | 5 |
| Hg | Serum haemoglobin | Numerical |  |
| Sz | Tumour size | Numerical |  |
| Sg | Stage/histolic grade combination | Numerical |  |
| Ap | Serum prostatic acid phosphase | Numerical |  |
| bm | Bone metastasis | Present/absent | 1 |
| Sdate |  |  |  |
| Pf.coded |  |  |  |

# Data is tidied and explored

Numerous steps were performed to tidy the data, for example joining levels to reduce categories.

It was also worth examining the data to make sure for example each treatment had similar patient number and were not statistically under represented.

# Determining complexity of model

To prevent overfitting, the complexity of the model can be ascertained based on the number of events that occurred. For example we can see that there were 502 patients in this experiment, however number of events occurred (deaths in given time) was 354. Therefore the number of parameters that can be used in this model is

\* divide by 20 for a more strict assessment

# number alive and dead

> prostate %>% count(status)

status n

1 alive 148

2 dead - prostatic ca 130

3 dead - heart or vascular 96

4 dead - cerebrovascular 31

5 dead - pulmonary embolus 14

6 dead - other ca 25

7 dead - respiratory disease 16

8 dead - other specific non-ca 28

9 dead - unspecified non-ca 7

10 dead - unknown cause 7

> sum(prostate$status != 'alive', na.rm=TRUE)

[1] **354**

# Predict missing data using transcan() function

Transcan() function is part of the rms package which imputes missing values based on other predictors.

Takes each of the predictors that are found in the formula, to create a model that holistically incorporates all the variables to predict missing values.

w = transcan(~ sz + sg + ap + sbp + dbp + age + wt + hg + ekg + pf + bm + hx,

imputed = TRUE, trantab=TRUE, data = prostate, pl = FALSE, pr = FALSE)

To get accurate/informed predictions this assumes there is sufficient correlation between variables. To check for this need to look at **summary of Transcan function**. This produces a lot of information including missing values, R2, adjusted R2 and coefficient of canonical variants

> **summary(w)**

The adjusted R­2 value says how strong is the correlation of each of the variables to the others. As shown below we can deduce that age has a very low correlation with the other predictors, whereas sg and ap are strongly correlated with everything else. Therefore from a predicative power perspective, you could be more confident of the missing values would be more accurately achieved in sg and ap compared to age.

Adjusted R-squared:

sz sg ap sbp dbp age wt hg ekg pf bm hx

0.179 0.541 0.559 0.481 0.467 0.065 0.093 0.129 0.059 0.085 0.332 0.082

To determine where most of the predictive power for each variable is being inferred from then we look at the Coefficients of canonical variates (as shown below). Here we can see for example that with ***age*** is most strongly correlated with bm and least strongly correlated to ***sz***. Therefore when inferring missing values, this is most strongly associated to ***bm***.

*\*\*\* However because these values are normalised we must also consider the adjusted R2 value of age where we reported poor correlation, therefore predictability for this variable is a bit dodge\*\*\*\*\**

on the other hand, predictor ***ap*** has a high adjusted R2 value and missing values appear strongly associated with ***sg*** and ***bm***. We therefore can say that there is more confidence in inference of ***ap*** which is based most strongly on ***sg*** and ***bm***.

A picture containing text, receipt

Description automatically generated

## Imputing predicted values into data

To get imputed values as variables

attach(prostate)

sz = impute(w, sz, data = prostate)

sg = impute(w, sg, data = prostate)

age = impute(w, age, data = prostate)

wt = impute(w, wt, data = prostate)

ekg = impute(w, ekg, data = prostate)

To check whether this has worked call a column (ensure that the database has been added to global environment). It is then possible to see where missing values have been added where there is a an astrix for example in ekg we can see row 243 has been predicted as normal.

>ekg

Text

Description automatically generated with low confidence

We can also visually see this by plotting variables transformation. Missing values are shown by the ‘O’. This also informs how linear or non-linear the variables are.

We can see for example that age is non-linear and this is particularly important for continuous data. We would therefore consider investing certain degrees of freedom in describing this data, using cubic splines. (see later stages)

> ggplot(w,scale=TRUE)+theme(axis.text.x=element\_text(size=6))

Chart

Description automatically generated

# Create a datadist object

Create a datadist object. This stores all information about the dataframe including mean, median max, min value etc.

This function is a utility function which tells other functions in the rms library where to get information about that variable.

# Kaplan Meier

To understand treatment effect on survival a performance analysis was performed using **a Kaplan-Meier survival analysis**

The question being asked was how different treatment types affect patient survival, these being different concentrations of oestrogen, where Placebo was the control.

Chart, histogram

Description automatically generated# create survival object

S = Surv(dtime, status %nin% 'alive')

#fit kaplan-Meier

f.kap = npsurv(S~rx)

# Plot

survplotp(f.kap)

# perform logrank test

survdiff(S~rx)

N Observed Expected (O-E)^2/E (O-E)^2/V

rx=placebo 127 95 87.6 0.619 0.839

rx=0.2 mg estrogen 124 95 84.9 1.212 1.626

rx=1.0 mg estrogen 126 71 95.9 6.479 9.072

rx=5.0 mg estrogen 125 93 85.6 0.644 0.867

Chisq= 9.1 on 3 degrees of freedom, p= 0.03

Visually it can be seen that 1mg of oestrogen increased survival time compared to control. Whereas 0.2mg and 5mg appears to have no affect.

To test whether this was significant a logrank test can be performed. Here we can see there was a P-value of less than 0.05 meaning that there is evidence to suggest that there is a significant difference in survival between treatment types.

Note this assessment is only including a single parameter (treatment type), and ignore other variables and limit its descriptive power which may have significant affect to survival. To perform a more comprehensive assessment, a cox hazard proportional model would need to be performed which can incorporate numerous predictors.

\*\*\*\*\* essentially adjusted R2 gives the confidence of your predictability of variable whereas coefficient of variables denotes the proportion of each variable in its predictability. \*\*\*\*\*\*\*

There are other ways of predicting missing values using but this is more complicated and outside the scope of this.

# Perform cox model

Here we develop a cox model. As detailed earlier we have 354/15 =23 predictors to play with without overfitting model. We have also seen that some of the variables are non-linear so it is worth including restricted cubic splines **rcs()** to relax the linearity assumption as part of the cox model. Example below uses 4 knots for each cublic spline.

f = cph(S ~ rx + rcs(age, 4) + rcs(wt, 4) + pf + hx + rcs(sbp, 4) + rcs(dbp, 4) + ekg + rcs(hg, 4) + rcs(sg, 4) + rcs(sz, 4) + rcs(log(ap), 4) + bm)

print(f, latex = TRUE, coefs = FALSE)

Text

Description automatically generated

From this we can see that we have 37 degrees of freedom, so we are using too many parameters therefore we are likely overfitting the model.

This overfitting is demonstrated by looking at the shrinkage coefficient which estimates how much noise we are fitting.

## How much data reduction is necessary?

When determining how much data reduction to undergo The Frank Harrell book (pg 87) suggests to prevent overfitting a rough guide is to achieve a greater than 0.9 data shrinkage statistic (see below formula). The shrinkage statistic determines how much noise is being overfitted where 0 is 100% noise and 1 is 0% noise. Therefore if this falls below the 0.9 threshold then there is concern t with the lack of calibration the model may receive of different data

### Shrinkage coefficient

From this example we can see from shrinkage coefficient we can estimate that 27% is noise (overfitting)

***1 - 0.73 = 0.27***

***=27%***

\*\*\*\* ideally want shrinkage closest to 1 \*\*\*\*\*

What predictors are linear/nonlinear

### AIC

The AIC statistic is a way of comparing two models for suitability using the likelihood statistic but penalising for increased complexity. As such a preferential model is chosen based on an AIC value closer to 0.

### AIC is calculated by:

\*\*\*\*\*\* However is this an appropriate method of doing so????

# Improving the model - data reduction

When trying to develop a model more parsimonious, there is a trade off between bias and variance. By decreasing the number of predictors will Increase predicitive powers by decreasing variance in estimating parameter estimation, however this will increase bias by moving further away from the true model. Therefore to get optimal prediction function a balance between gain in variance and loss in bias is required.

A common way to achieve this is using a stepwise method, whereby the ‘*best selection*’ of predictors are chosen by iteratively removing predictors and comparing the test statistic, to achieve optimal or sub optimal fit. However this causes

An example of a stepwise method is shown below. This Method isst **not advisable** to fit a full model and then subsequently remove predictors to see whether this has reduced predictive power.

As shown in the below script it is tempting to remove age as a predictor as based on the P-value there has been no significant reduction in predictive power. However, this is dangerous because once you have fitted a model you have extracted information such as the test statistic with the Chi2 and when you compare against another model where you have removed predictors then you come up with a separate set of statistics that does not account that you have already fitted a model and obtained certain information. Not keeping

# full model

f.full = coxph(Surv(time,status) ~ data = prostate)

# reduced model

f.reduced = f.full = coxph(Surv(time,status) ~ - age, data = prostate)

loglik Chisq Df P(>|Chi|)

1 -381.83

2 -381.83 0 1 0.9999

\*\*\*\*alternatively using the MASS package stepAIC(f.full)

>anova(f.nearfull, f.full, test = "LRT")

## So how to choose predictors?

Prior to making the model you should already have a priori knowledge for including the predictor in the first place. i.e you should not delete predictors after already testing

## Reducing predictors by decreasing cubic spine knots

From the model it is worth exploring the data using the anova function

# original model

f = cph(S ~ rx + rcs(age, 4) + rcs(wt, 4) + pf + hx + rcs(sbp, 4) + rcs(dbp, 4) + ekg + rcs(hg, 4) + rcs(sg, 4) + rcs(sz, 4) + rcs(log(ap), 4) + bm)

# exploring data

anova(f)

Table

Description automatically generated

As we can see from the data there are certain predictors being entered into the model nonlinearly. For example age shows evidence of non-linearity, where there are three degrees of freedom two of which are non-linear with a p-value of 0.01. However by looking at the Chi-squared values, the predictive power of the non-linear predictors (8.95) is about half of that of the total predictor (16.66), thus the linear predictive power which comprises of 1 degree of freedom = 16.66-8.95 (7.71).

Another example would be sz which has 5 degrees of freedom. The linear part has 3 degrees of freedom (5-2=3) and a high predicitive power whereas the non-linear part which has two degrees of freedom and a low predicative power (1.66). Arguably this is wasting the 2 degrees of freedom and there could be an argument to assume that the variable enters the model linearly or reducing cubic splines from 4 to 3.

As shown below.

fx = cph(S ~ rx + rcs(age, 3) + rcs(wt, 3) + pf + hx + rcs(sbp, 3) + rcs(dbp, 3) +

ekg + rcs(hg, 3) + rcs(sg, 3) + rcs(sz, 3) + rcs(log(ap), 5) + bm)

Table

Description automatically generated

\*\*\*\*note that ap has a high predictive power for non-linearity . Should this predictor therefore be removed?????

We can compare the new models by assessing the AIC and shrinkage coefficient

# Get test statistic of new model

Print(fx)

Text

Description automatically generated

Using the equations

From this we can see whilst there has been improvement to the model where AIC has increased from 63.02 to 64.07, the improvements are rather small. Furthermore by looking at the shrinkage coefficient we can estimate that there is still 25% noise being plotted in the model, thus implicating overfitting (remember ideally want shrinkage to be > 0.9).

# Visualising the predicted model

Based on the on the fitted model it is possible to plot the predicted effect from each model using the predict function()

ggplot(predict(f2, sepdiscrete = ‘vertical’, nlevels=4, vnames= ‘names’))

# Graphical user interface, diagram, application Description automatically generated

# Analysing interactions and comparing to predicted model

If you assume there is an interaction between two variables, then you would multiply the two predictors. This was demonstrated in the case study where treatment dosage was assessed for interaction between all other variables.

This interaction was compared using the predicted model, meaning that if interactions are not found then they do not need to be included into the model and does not jeopardise the ‘honesty’ of the model development.

# create model, **Note the inclusion of X = True, Y=true**

f = cph(S ~ rx + rcs(age, 4) + rcs(wt, 4) + pf + hx + rcs(sbp, 4) + rcs(dbp, 4) +

ekg + rcs(hg, 4) + rcs(sg, 4) + rcs(sz, 4) + rcs(log(ap), 4) + bm,

x = TRUE, y = TRUE, surv = TRUE, time.inc = 5\*12)

Table

Description automatically generated# Return predicted regressors multiplied by respective coefficient

z = predict(f, type = 'terms')

# get treatment column (rx)

z.dose = z[, 1] # z[,1] get column rx (dose)

# get all columns apart from rx

z.other = z[, -1]

# compare model. Multiplying rx with all other predictors to determine possible interactions

f.ia = cph(S ~ z.dose \* z.other)

#see individual interaction results

print(f.ia)

#see global interaction results

anova(f.ia)

Text

Description automatically generated with medium confidence

## Interpretation interactions

A low Wald statistic and high P-values indicates there is no evidence to suggest that oestrogen dosage does not interact with the other predictors. Note that **pf** does have a ‘*significant*’ P-value, however due to the models complexity, high number of parameters incorporated and the limited number of events, this tends to exaggerate significance. It is therefore reasonable to ignore this, and indicate a significance when p-value is 0.005-0.0005. We also see from the global a p-value of 0.072 which again is outside the significance threshold (which again is anti-conservative)

# Checking Model Assumptions

## Are the variables proportional?

To do this firstly need to produces a matrix where values observed in the data are multiplied by the coefficients estimated by the model using the predict() function. The type = ‘terms’ ensures you are multiplying coefficient to the different predictors.

Z = predict(f2, type = ‘terms’)

Next step is to create a cox model using the predicted values (note this is done using 1df). Note the inclusion of x=true and y=true as this stores the outputs, which will be used in the later steps for Schoenfeld residuals

f.short = cph(S ~ z, x = TRUE, y= TRUE)

Perform Shoenfelds residuals test. This is achieved by transform = ‘identity’ other types of test can also be used. Also ensure that **terms = false** as this performs an analysis on each single predictor rather than on models entirety.

phtest = cox.zph(f.short, transform = 'identity', terms = FALSE)

phtest

A picture containing text, receipt

Description automatically generated

From the results we can see that rx is < 0.05 which suggests that there is a deviation from the constant line. However because of the complexity of this model and small sample size the 0.05 threshold is arguably too small. Furthermore when we look at the Global p-value, this suggest the model as a whole does not validate the proportional assumption.

It is possible to observe this visually by plotting the residuals where we see a slight upwards trend in rx. For this to be proportional we would expect the line be horizontal and in line with the predicted variables coefficient as shown in **phtest**. This means that effect is proportional over time ie in line with baseline hazard. Therefore we would want the coefficient to be within the confidence interval throughout.

plot(phtest, var = 'rx')

Chart, scatter chart

Description automatically generated

In the example below we are plotting residuals to variable ‘ap’, however this can be subsitituted for any variable

# Describing Predictor effects

It is possible visualise the impact of each variable related to the **log hazard** of event occurring (death) by plotting the predicted values from the model. The predict function also provides a 95% confidence interval.

This has been coded to distinguish between continuous (top) and discrete variables (bottom)

ggplot(Predict(f2), sepdiscrete='vertical', nlevels=4, vnames='names')

Graphical user interface, diagram, application

Description automatically generated

As shown in the data we can see for the predictor **age** log hazard increases as they get older, which is particularly prominent after age of 70, while **ap** remains relatively constant throughout.

# Validating model

After developing a model it is important to validate it to ensure that the predicted values accurately predict responses on future subjects not used in the development of the data.

There are two types of model validation

**Internal:** where you are comparing model with data collected from original sample either through data splitting (downside being reduced precision in original model) or resampling using bootstrapping, which yields less biased results

**External:** This tests model against an external dataset, with independent subjects. Data splitting is a form of external validation where the data is split for i) *training* the model and ii) *testing*/validating model. However this method reduces sample size by reducing the number of subjects used in the initial development thus reducing the precision and power of the model and limiting the number of parameters being able to be used.

Because of the disadvantages described above data splitting it is more arguably more appropriate to resample data using bootstrapping (internal validation)

Here Somer’s Dxy rank correlation is employed to compare **observed survival** and **predicted log hazard** – Here we can obtain concordance index among other statistics**.** Bootstrapping is used to prevent overfitting as this repeats model *n* number of times.

# set seed generates a predictable set of random numbers for repeatable results

set.seed(1)

# Validating the f2 model using bootstrapping of 300

v = validate(f2, B=300)

Text

Description automatically generated

## Interpretation

### C-index

Assessing a models goodness of fit can be determined by the Concordance index (C-index) which is ranks risk score against observed. This can be achieved by transforming the Dxy value by the following formula:

Therefore:

This suggests the model is **not performing** very well as the model correctly ranks around 63% of the subjects. **Ideally this should be > 0.8**

### Slope

The slope gives the approximation of the shrinkage of the model (The proportion of the data being over fitted). Here we see that shrinkage has been calculated at **0.796 (**where previously this was estimated at 0.75)**.**

Here we say the model is fitting ~20% noise (1-0.796\*100), where ideally this should be less than 10% (SHRINKAGE INDEX 0.9)

\*\*\*\* Note the importance of including a validation test rather than relying on assessing this based on likelihood statistic such as AIC. As we have seen through validation that the model is not preforming very well.

# Calibration

Calibration compares how closely the expected and observed are (this is different to validation, which assess the rank order).

Expected (x-axis) is calculated from the Kaplan-Meier survival estimate survival while predicted (y-axis) is the estimated survival at given point in time as denoted by the *time.inc* argument in the model and *m* argument in calibrate function. Note that the fitted model must include specified arguments *x=TRUE* and *y=TRUE*.

**\*\*\*\*\* Note Bootstrapping should ideally be around 500-1000**

# set random seed

set.seed(1)

# set time interval where survival will be measured against, Note this was set before but for clarity we are reinstating this at 60 months (5years)

f2 = update(f2, time.inc = 60)

Chart, line chart

Description automatically generated# calibrate data set using bootstrap of 300 permutations (b), point in time to calibrate model 60 months (u) – **this needs to be the same as *time.inc****,* maximum dimensions (m) – not sure if this is number of predictors used?

cal = calibrate(f2, B=300, u=60, maxdim = 4)

# Plot results

plot(cal, subtitles = FALSE) +

legend(0.55,0.1, legend = c('expected', 'observed'), col=c("black", "blue"), lty = 1, cex=0.8)

#add lines to plot to visualise over estimation of survival

abline(h=0.5, col="grey", lty = 2, cex = 1.5)

Chart, line chart

Description automatically generatedabline(v=0.54, col = 'grey', lty = 2, cex = 0.5)

# Include confidence intervals for the estimates of fraction surviving

cal = calibrate(f2, cmethod = 'KM', u = 5\*12, B = 150)

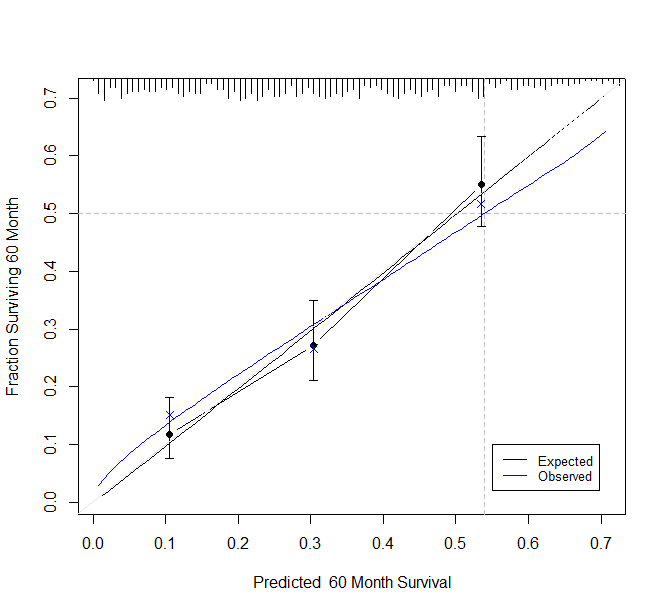
plot(cal, add = TRUE)

## Interpretation

Looking at the median at (50% surviving at 5 years), we see that the model over predicts survival (~54%). **This is dangerous for a survival analysis model**. Ideally we would want this to be consistent where expected and observed are the same.

The confidence intervals show that with the variability of the predicted values are statistically within the variability of the observed survival. Therefore it can be argued that the overestimation is not significantly significant.

It is also worth while looking at this at different survival time points like for example 24 months. Here we see that the again using the median survival fraction the model tends to **underestimate** survival (~45%).



# calibrate at 24 months survival

f2 = update(f2, time.inc = 24)

cal = calibrate(f2, B=300, u=24, maxdim = 4)

plot(cal, subtitles = FALSE) +

abline(h=0.5, col="grey", lty = 2, cex = 1.5)

abline(v=0.455, col = 'grey', lty = 2, cex = 0.5) +

legend(0.7,0.3, legend = c('Expected', 'Observed'), col=c("black", "blue"), lty = 1, cex=0.8)

# Summarising data

Data can be presented to show how each variable effects log hazard. This compares log hazard at different interquartile ranges, for example age 70 and age 76.

Chart

Description automatically generated

Cox hazard proportional model pipeline

1. Tidy data
2. Impute missing values
3. Kaplan Meier

Pipe line