```
pacman::p_load(
  tidyverse,
  lmtest,
  sandwich,
  broom,
  VIM,
  splines,
  rmutil
)

# load some data, select variables:
# set the seed for reproducibility
set.seed(12345)

## generate the observed data
n=1000
# uniform random variable bounded by 0 and 8
x = runif(n,0,8)
# continuous outcome as a complex function of x
y = 5 + 4*sqrt(9 * x)*as.numeric(x<2) +
as.numeric(x>=2)*(abs(x-6)^(2)) + rlaplace(n)

a <- data.frame(x=x,y=y)
head(a)

# linear fit
model1 <- glm(y ~ x)
a$y_pred1 <- predict(model1)

ggplot(a) +
  geom_point(aes(y=y,x=x),color="gray") +
  geom_line(aes(y=y_pred1,x=x), color="red")

## key arguments for ns:
## df   = degrees of freedom. One can supply df rather than knots;
## ns() then chooses df - 1 - intercept knots at suitably chosen
## quantiles of x (which will ignore missing values). The default,
## df = NULL, sets the number of inner knots as length(knots).
##
## knots        = breakpoints that define the spline. The default is
no knots;
## together with the natural boundary conditions this results in a
## basis for linear regression on x. Typical values are the mean or
## median for one knot, quantiles for more knots. See also
Boundary.knots.

## key arguments for bs:
## df   = degrees of freedom; one can specify df rather than knots;
## bs() then chooses df-degree (minus one if there is an intercept)
## knots at suitable quantiles of x (which will ignore missing
```

```
values).
## The default, NULL, takes the number of inner knots as
length(knots).
## If that is zero as per default, that corresponds to df = degree -
intercept.
##
## knots = the internal breakpoints that define the spline. The
default is
## NULL, which results in a basis for ordinary polynomial regression.
Typical
## values are the mean or median for one knot, quantiles for more
knots.
## See also Boundary.knots.
##
## degree       = degree of the piecewise polynomial—default is 3 for
cubic splines.

# ns
model2 <- glm(y ~ ns(x, df = 5))
a$y_pred2 <- predict(model2)

ggplot(a) +
  geom_point(aes(y=y,x=x),color="gray") +
  geom_line(aes(y=y_pred1,x=x), color="red") +
  geom_line(aes(y=y_pred2,x=x), color="blue",
            linetype = "dashed")

# bs
model3 <- glm(y ~ bs(x, df = 5))
a$y_pred3 <- predict(model3)

ggplot(a) +
  geom_point(aes(y=y,x=x),color="gray") +
  geom_line(aes(y=y_pred1,x=x), color="red") +
  geom_line(aes(y=y_pred2,x=x), color="blue",
            linetype = "dashed")  +
  geom_line(aes(y=y_pred3,x=x), color="black")




# ns
model2a <- glm(y ~ ns(x, knots = c(1,2,4,6)))
a$y_pred2 <- predict(model2a)

ggplot(a) +
  geom_point(aes(y=y,x=x),color="gray") +
  geom_line(aes(y=y_pred1,x=x), color="red") +
  geom_line(aes(y=y_pred2,x=x), color="blue",
```

```
            linetype = "dashed")

# bs
model3a <- glm(y ~ bs(x, knots = c(1,2,4,6), degree = 4))
a$y_pred3 <- predict(model3a)

ggplot(a) +
  geom_point(aes(y=y,x=x),color="gray") +
  geom_line(aes(y=y_pred1,x=x), color="red") +
  geom_line(aes(y=y_pred2,x=x), color="blue",
            linetype = "dashed")  +
  geom_line(aes(y=y_pred3,x=x), color="black")

## comparing models
##
## if the models are nested

lrtest(model1,model2)

lrtest(model1,model3)

## if the models are not nested, pick the lower AIC

AIC(model1)
AIC(model2)

AIC(model2)
AIC(model3)
```