周课 7

陈漂亮 Savvy 2020

目录

1. Non-Parametric Model

- Penalized likelihood;
- smoothing;
- fitting wiggly lines through points;
- semi-parametric models;

- splines

$$
\begin{aligned}
& Y_i \sim \pi(\lambda_i, \theta) \\
& g(\lambda_i) = X_i\beta + f(W_i) \\
\text{Where, } & \bullet\, Y_i \text{ : response} \\
& \bullet\, \pi(\lambda_i, \theta) \text{ is the response distribution} \\
& \bullet\, X_i, W_i are covariates \\
& \bullet\, f(w) : \text{ is the smoothing function} \\
& \bullet\, g(\lambda) : \text{ is the link function} \\
& \bullet\, \beta : \text{ coefficients}
\end{aligned}
$$

1. Penalized Likelihood

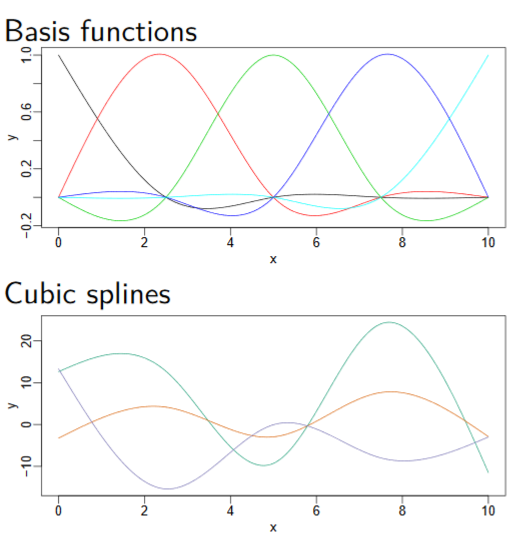
$$
\begin{aligned}
L_P(\beta, f, \alpha; Y) = \log(\pi(Y; \beta, f)) - \alpha \int \left[\frac{\partial^2 f(w)}{\partial w^2}\right]^2 du \\
\text{Where, } \bullet\, \alpha\text{: penalty parameter, } \alpha \uparrow \implies \text{ smoother } f \\
\bullet\, \text{smoother } f(x) \implies \text{ smaller } f''(x) \\
\implies \hat{\beta}(\alpha), \hat{f}(\alpha) = \arg\min_{\beta, f} L_P(\beta, f, \alpha; Y)
\end{aligned}
$$

- A good $\hat{f}$ is a compromise between fitting the data and being smooth.

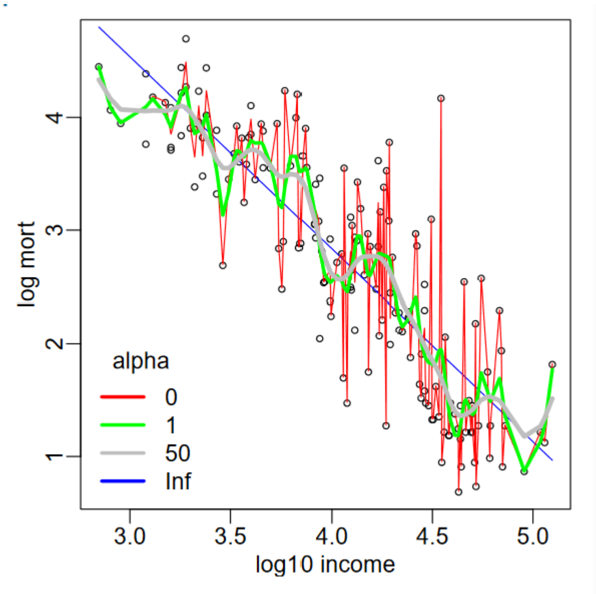
1.2 Cubic Spline

- The f that maximizes the penalized likehood must be a cubic spline polynomial…

```
knitr::include_graphics("1.png")
```



```r
knitr::include_graphics("2.png")
```



- The basis function of cubic splines: $ax^3 + bx^2 + cx + d$...

  Maiximizing likelihood over all possible f:

- The larger the α, the smoother the curve (f)...

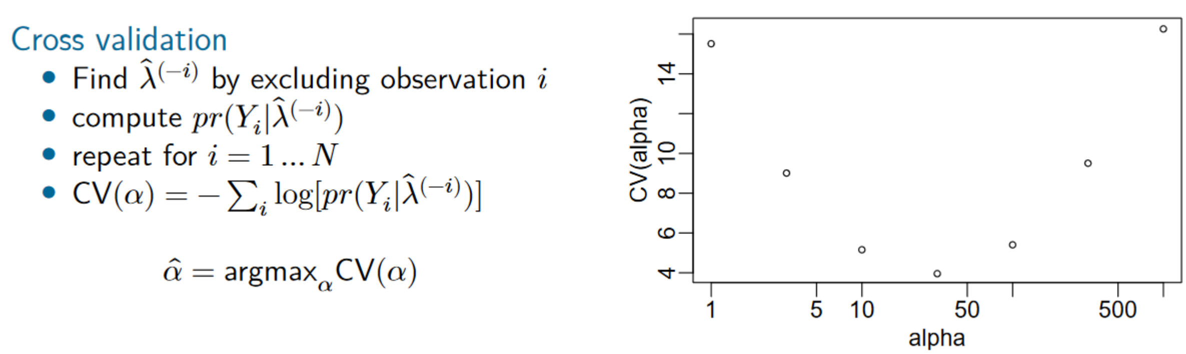
- When $\alpha \to \infty$, f is a straight line.

  How?:

- Divide your data (evenly) into K subsets, and fit a cubic spline on each subset. Make sure the f function is continous/1st-order-diff/2nd-order-diff at each knot...

  Choosing α: Cross Validation

```r
knitr::include_graphics("3.png")
```



2. Generalized Additive Model (GAM)

- Fit a GAM for the Math score data…

```
library('mgcv')
```

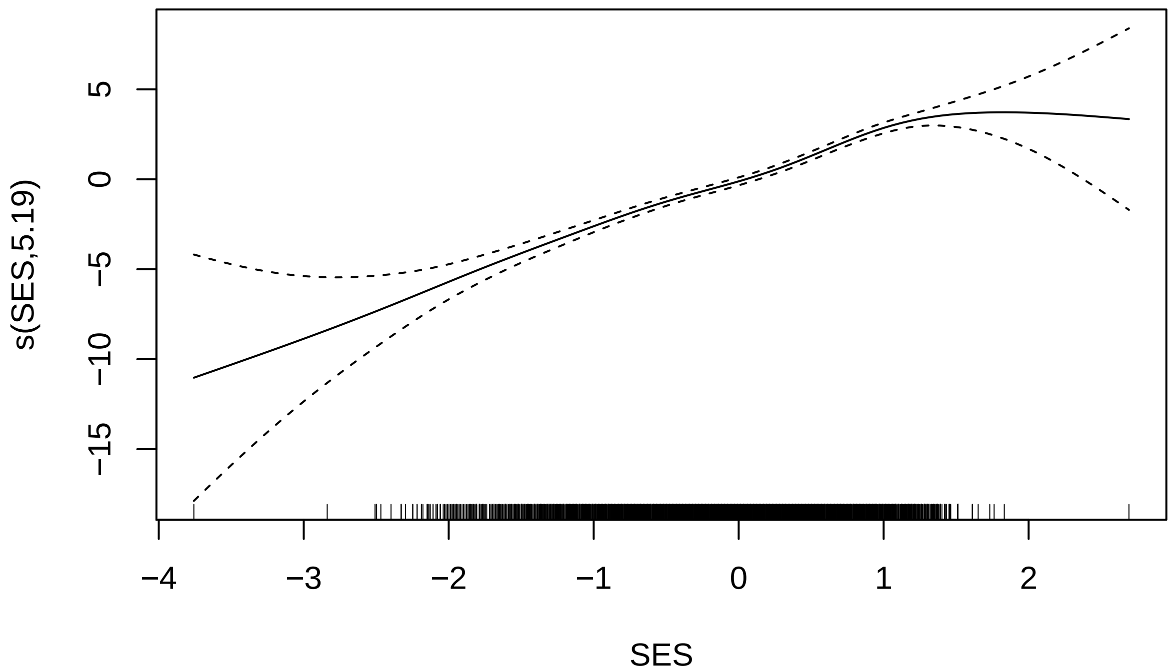
```
## Loading required package: nlme
```

```
## This is mgcv 1.8-28. For overview type 'help("mgcv-package")'.
```

```
mathGam =gam(MathAch~s(SES)+Minority*Sex,
             data=MathAchieve)
knitr::kable(summary(mathGam)$p.table[,1:2],
             digits=1)
```

	Estimate	Std. Error
(Intercept)	14.3	0.1
MinorityYes	-2.9	0.2
SexFemale	-1.4	0.2
MinorityYes:SexFemale	0.2	0.3

```
plot(mathGam)
```




```
mathGam$sp # smoothing parameter
```

```
##    s(SES)
## 0.8254378
```

2.1 Smoothing Interation

- Now we fit another GAM, with interaction between the covariates that are being smoothed…

```
mathGamInt =gam(MathAch~s(SES,by=Minority)
                +Minority*Sex,
                data=MathAchieve)
knitr::kable(summary(mathGamInt)$p.table[,1:2],
             digits=1)
```

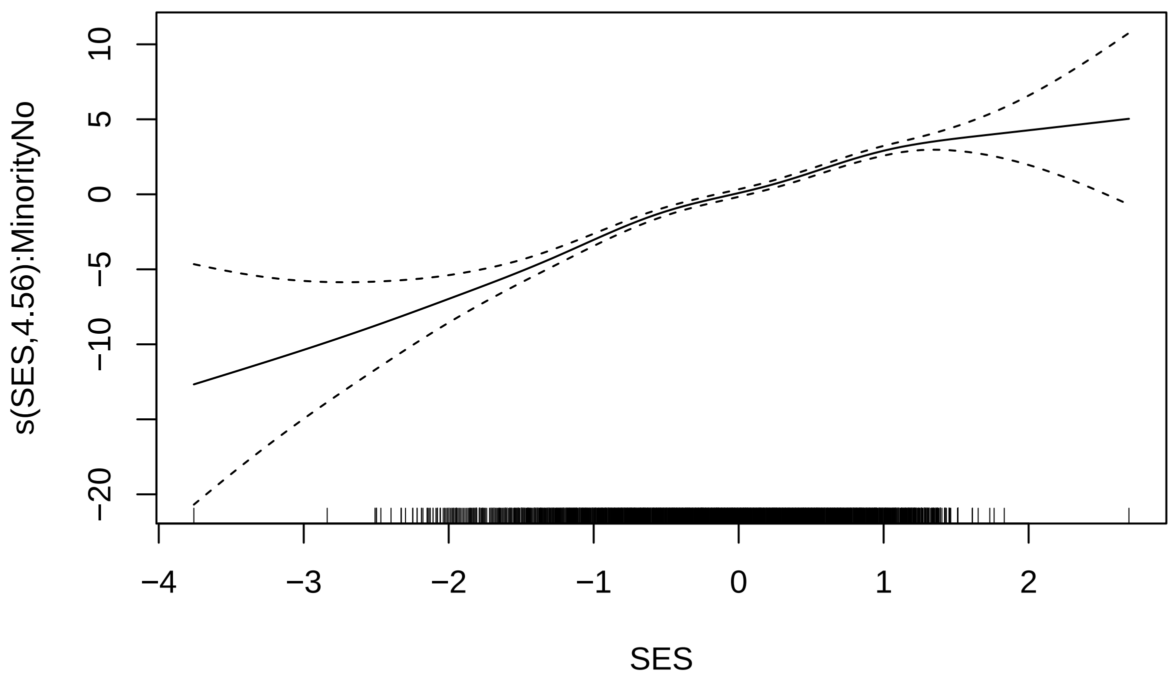
	Estimate	Std. Error
(Intercept)	14.2	0.1
MinorityYes	-3.0	0.3
SexFemale	-1.4	0.2

	Estimate	Std. Error
MinorityYes:SexFemale	0.1	0.3

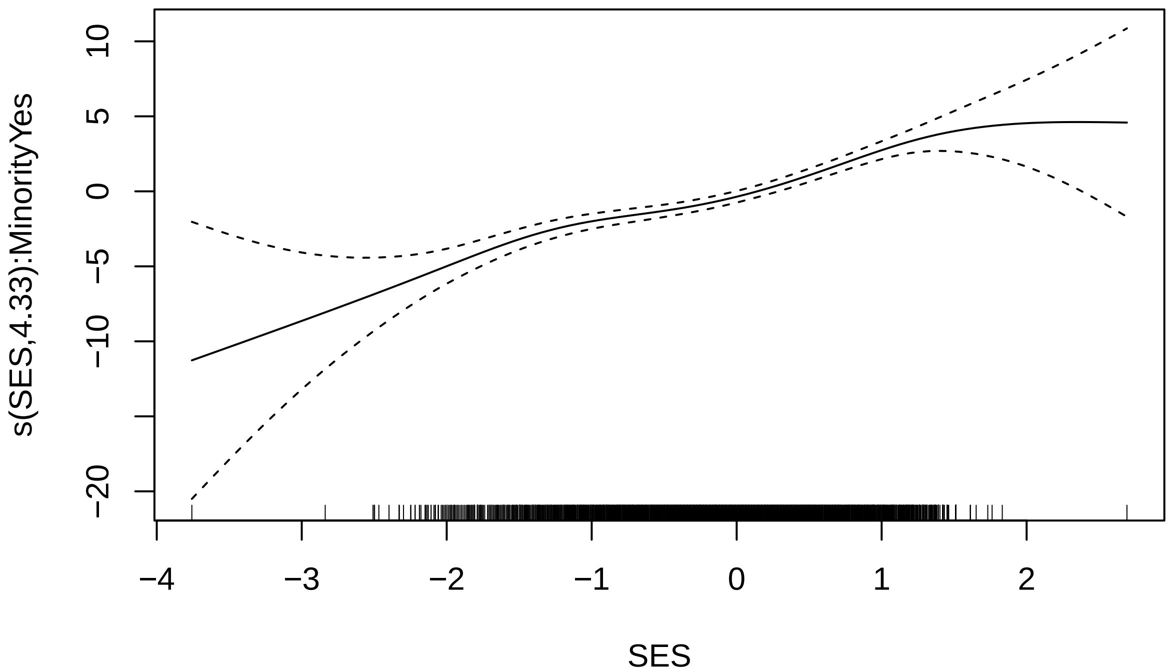
```
mathGamInt$sp
```

```
##  s(SES):MinorityNo s(SES):MinorityYes
##           0.820158           0.614983
```

```
# plot the SES/minority
plot(mathGamInt,select =1)
```




```
plot(mathGamInt,select =2)
```



2.2 Common smoothing parameter

```
knitr::include_graphics("4.png")
```

$$\begin{aligned} Y_{ij} \sim & N(\lambda_{ij}, \tau^2) \\ \lambda_{ij} = & X_{ij}\beta + f_i(W_{ij}; \nu) \end{aligned}$$

- Y_{ij} is the observation for individual j in ethnic group i
- X_{ij} is a vector of covariates (ethnic group, sex, interaction)
- $f_i(w; \nu)$ is the smoothly-varying function of SES
    - for ethnic group i
    - with roughness parameter ν.

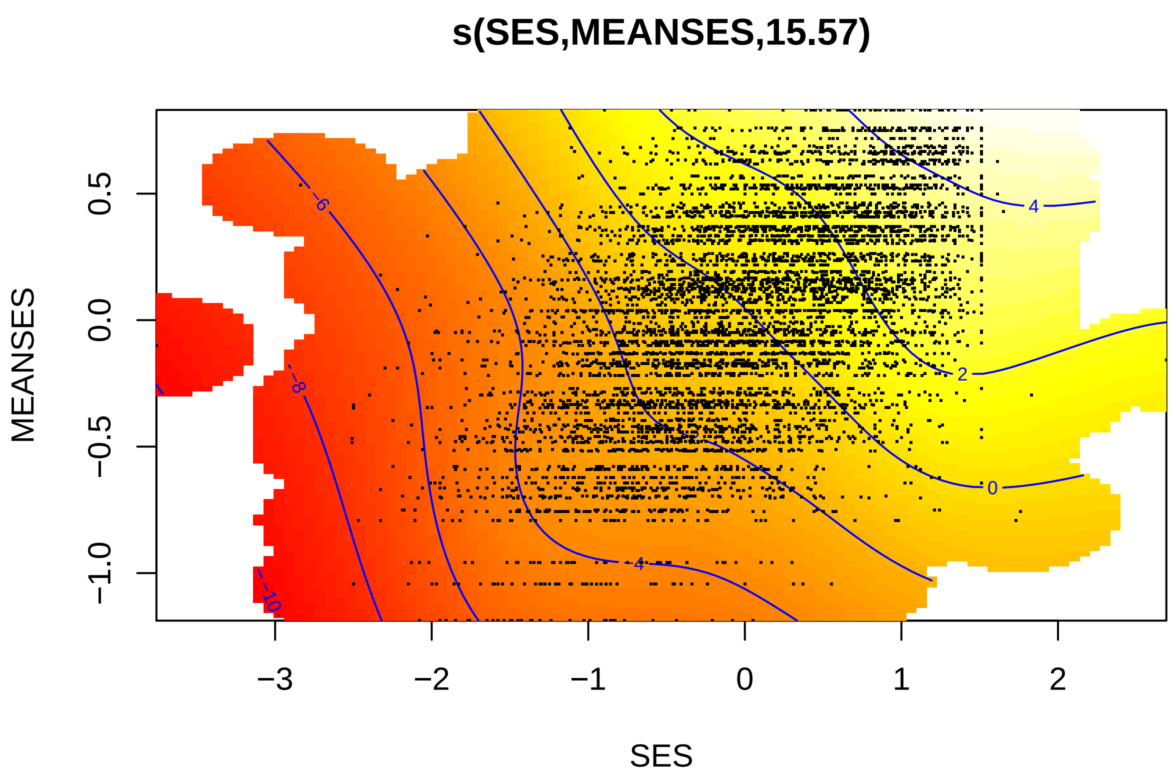
```
mathGamIntC =gam(MathAch~s(SES,by=Minority,id=1) +Minority*Sex,
                 data=MathAchieve)


mathGamIntC$sp
```

```
## s(SES):MinorityNo
```

```
## 0.7492505
```

2.3 2-D smoothing

```
mathGam2 =gam(MathAch~s(SES, MEANSES)+Minority*+Sex,
              data=MathAchieve)
plot(mathGam2,scheme =2,n2 =100)
```



- If you are from upper class, your score is still likely higher even if your school is weaker…

2.4 Poisson GAM: Ontario deaths

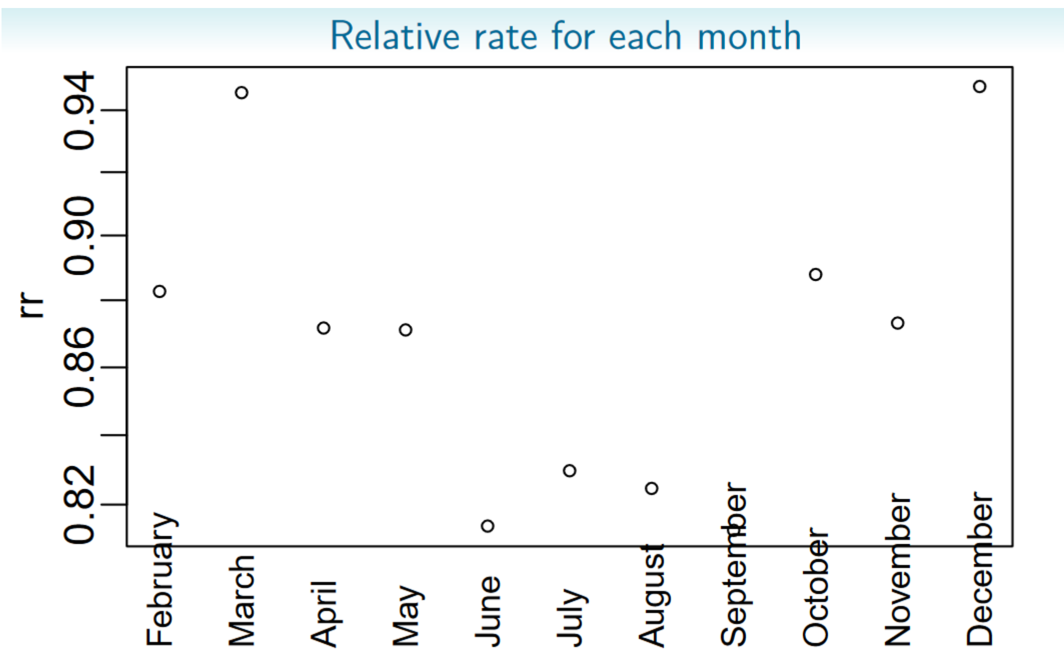
$$Y_i \sim Poisson(\lambda_i)$$
$$\log(\lambda_i) = X_i\beta + f(time)$$

where, • λ_i is the relative rate of death in the ith month

```
deathsGam =gam(+Value~month+s(timeNumeric),
              data=oDeaths,family='poisson'+)
```

```
knitr::kable(summary(deathsGam)$p.table[,1:2],
             digits=3,col.names=c('est','se'))
```

	est	se
(Intercept)	9.001	0.002
monthFebruary	-0.124	0.003
monthMarch	-0.055	0.003
monthApril	-0.137	0.003
monthMay	-0.138	0.003
monthJune	-0.205	0.003
monthJuly	-0.186	0.003
monthAugust	-0.192	0.003
monthSeptember	-0.207	0.003
monthOctober	-0.118	0.003
monthNovember	-0.135	0.003
monthDecember	-0.053	0.003



- relative rate: relative to the baseline, i.e. Janurary…
- Note that different month has different number of days, so, offset!!

$$Y_i \sim Poisson(O_i\lambda_i)$$

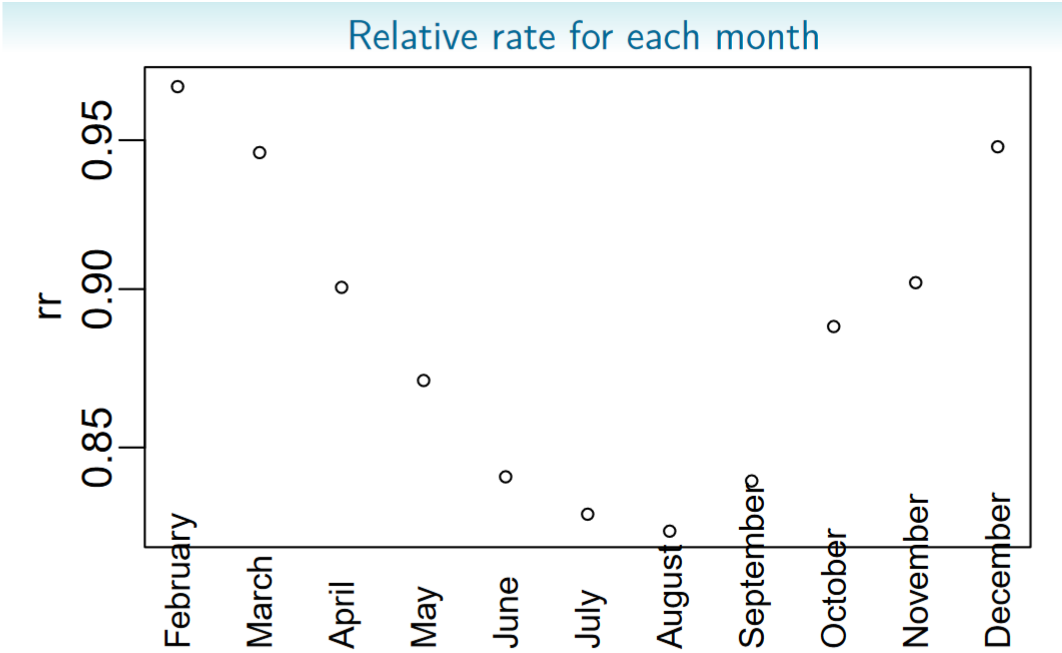
$$\log(\lambda_i) = X_i\beta + f(time)$$

where, • λ_i is the relative rate of death in the ith month

• O_i is the offset term

```
deathsGam =gam(Value~month+s(timeNumeric)
               +offset(nDays),
               data=oDeaths,
              family='poisson')
knitr::kable(summary(deathsGam)$p.table[,1:2],
             digits=3,col.names=c('est','se'))
```

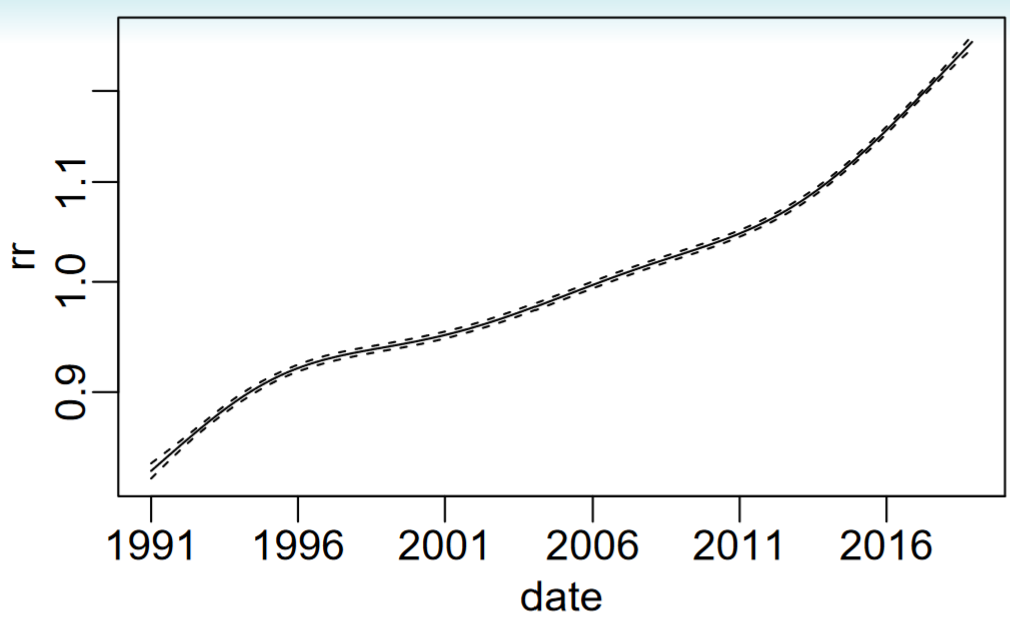
	est	se
(Intercept)	5.567	0.002
monthFebruary	-0.031	0.003
monthMarch	-0.055	0.003
monthApril	-0.104	0.003
monthMay	-0.138	0.003
monthJune	-0.173	0.003
monthJuly	-0.186	0.003
monthAugust	-0.192	0.003
monthSeptember	-0.174	0.003
monthOctober	-0.118	0.003
monthNovember	-0.102	0.003
monthDecember	-0.053	0.003



2.5 Prediction

2.5.1 Trend

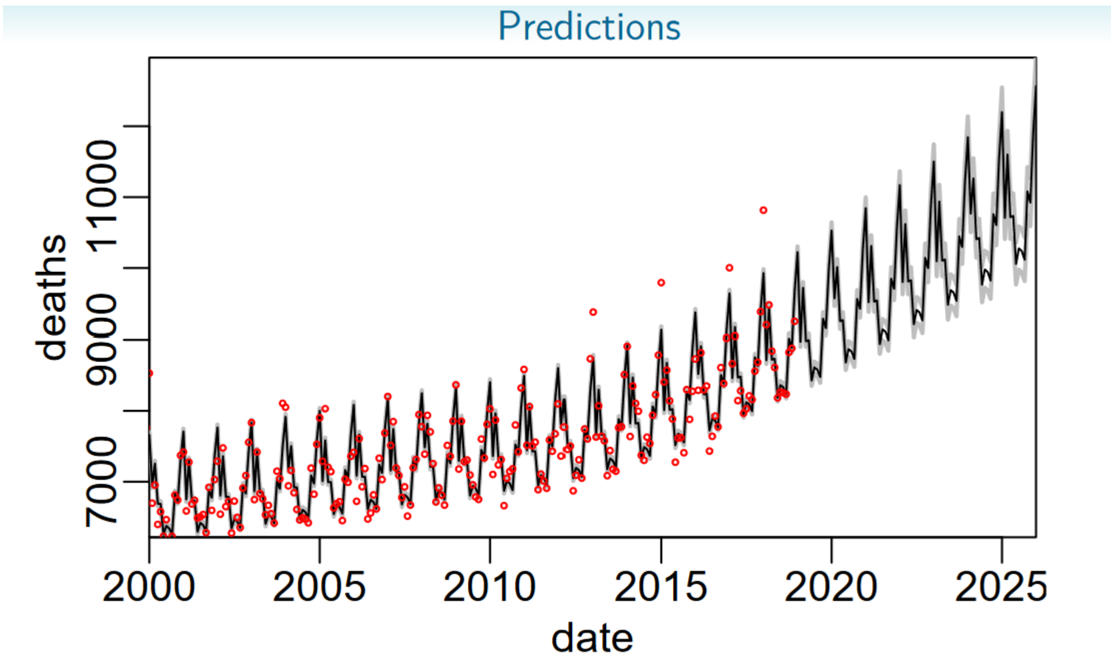
```
dSeq =seq(from =min(oDeaths$date),
          by ="5 years",length.out =10)
deathPred =as.matrix(as.data.frame(predict.gam(deathsGam, oDeaths,type ="terms",terms ="s(timeNume

deathPred =exp(deathPred%*%Pmisc::ciMat())
matplot(oDeaths$timeNumeric, deathPred,log ="y",
        xaxt ="n",xlab ="date",type ="l",
        lty =c(1,+2,2),col ="black",
        ylab ="rr")
axis(1,at =difftime(dSeq,
                    timeOrigin,units ="days"),
     labels =format(dSeq,"%Y"))
```



2.5.3 Forcasting

```
Stime =seq(from =as.Date("2000/1/1"),
           to =as.Date("2026/1/1"),
           by ="months")
newX =data.frame(timeNumeric
            =as.numeric(difftime(Stime,
                                 timeOrigin,
                                 units ="days")),
            month =months(Stime),
            nDays =log(Hmisc::monthDays(Stime)))
deathsPred =predict(deathsGam, newX,se.fit =TRUE)

deathsPred =as.data.frame(deathsPred)
deathsPred$lower =deathsPred$fit-2*deathsPred$se.fit
deathsPred$upper =deathsPred$fit+2*deathsPred$se.fit
matplot(Stime,
        exp(deathsPred[,c("lower","upper",+"fit")]),
        type ="l",lty =1,
        col =c("grey",+"grey","black"),
        lwd =c(2,2,1),xlab ="date",
        ylab ="deaths",yaxs ="i",xaxs ="i",
        xaxt ="n")
forAxis =seq(from =as.Date("2000/1/1"),
```

```
            to =as.Date("2026/1/1"),
            by ="5 years")
axis(1,as.numeric(forAxis),format(forAxis,"%Y"))
points(oDeaths$date,
       oDeaths$Value,cex =0.5,col ="red")
```



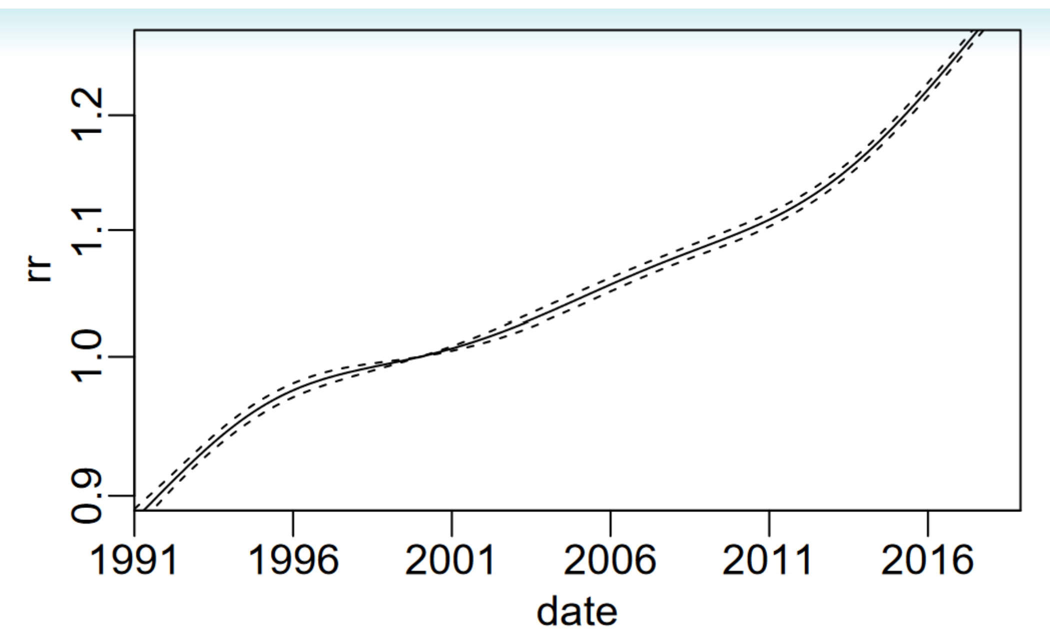
2.6 Change the Parameter constraint

- Add a constant to f(x) doesn't change the penalty;
- By default, f(x) sums to 0;
- But we don't know where does f(x) = 0;
- An alternative is to set $f(x_0) = 0$...

```
deathsGamC =gam(Value~month+s(timeNumeric,pc=0)
                +offset(nDays),
                data=oDeaths,+family='poisson')

# pc: param constraint

deathPredC =as.matrix(as.data.frame(predict.gam(deathsGamC,oDeaths,
                                                type ="terms",
```

```
                                                    terms ="s(timeNumeric)",
                                                    se.fit =TRUE)))

deathPredC =exp(deathPredC%*%Pmisc::ciMat())
matplot(oDeaths$timeNumeric, deathPredC,log ="y",
        xaxt ="n",xlab ="date",type ="l",
        lty =c(1,+2,2),col ="black",ylab ="rr",
        xaxs ="i",+yaxs ="i",ylim =c(0.89,1.28))
axis(1,at =difftime(dSeq, timeOrigin,
                    units ="days"),
     labels =format(dSeq,"%Y"))
```

	est	se
(Intercept)	5.510	0.003
monthFebruary	-0.031	0.003
monthMarch	-0.055	0.003
monthApril	-0.104	0.003
monthMay	-0.138	0.003
monthJune	-0.173	0.003
monthJuly	-0.186	0.003
monthAugust	-0.192	0.003
monthSeptember	-0.174	0.003
monthOctober	-0.118	0.003
monthNovember	-0.102	0.003
monthDecember	-0.053	0.003

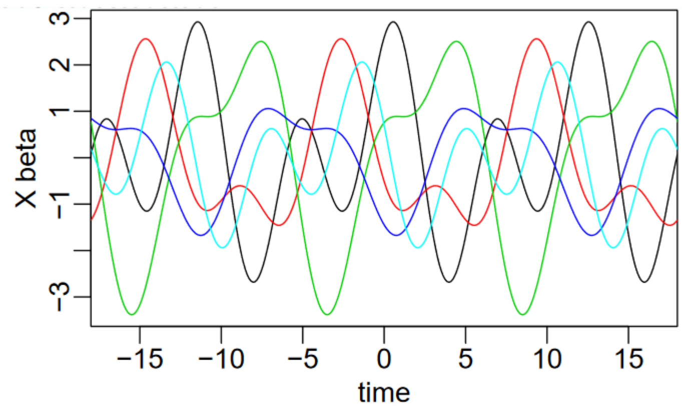


2.7 Modelling Seasonality

We can see clear seasonality from the month-effect plot.

- The trick to model seasonality is apply trignomitric function as the basis functions...

- The monthly effect isn't perfectly sinusoidal
- use a 12 month and a 6 month frequency

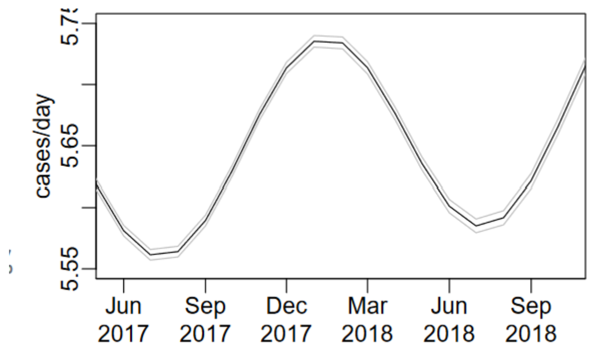


$$
\begin{aligned}
Y_i &\sim \text{Poisson}(O_i\lambda_i)\\
\log(\lambda_i) &= X_i\beta + f(t_i)\\
X_{i0} &= 1\\
X_{i1} &= \cos(2\pi t_i/12)\\
X_{i2} &= \sin(2\pi t_i/12)\\
X_{i3} &= \cos(2\pi t_i/6)\\
X_{i4} &= \sin(2\pi t_i/6)
\end{aligned}
$$

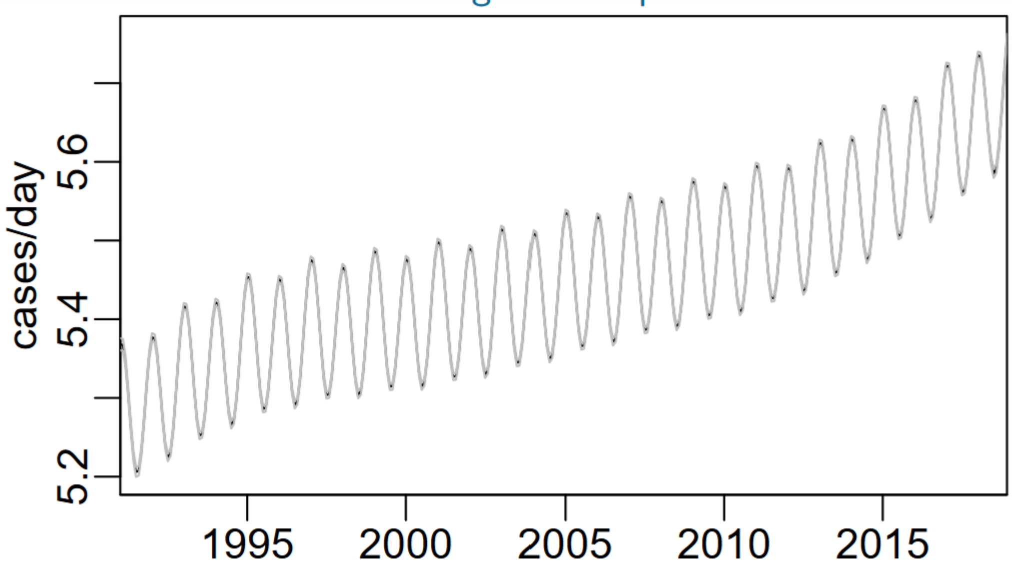
cos12	1.26	0.41	-0.01	-1.15	0.25
sin12	-0.33	-1.54	2.40	-0.29	-0.89
cos6	1.33	-0.93	0.76	-0.30	0.44
sin6	1.27	-0.29	-0.80	-0.41	-1.24

```
oDeaths$timeYears =oDeaths$timeNumeric/365.25
oDeaths$cos12 =cos(2*pi*oDeaths$timeYears)
oDeaths$sin12 =sin(2*pi*oDeaths$timeYears)
oDeaths$cos6 =cos(2*pi*oDeaths$timeYears/2)
oDeaths$sin6 =sin(2*pi*oDeaths$timeYears/2)

deathsGamS =gam(Value~cos12+sin12+cos6+sin6
              +s(timeNumeric)+offset(nDays),
              data=oDeaths,family='poisson')
knitr::kable(summary(deathsGamS)$p.table[,1:2],

# Predicting the seasonality
deathGamPred =predict(deathsGamS,cbind(oDeaths[,c("sin12","cos12","sin6",
                                                  "cos6","timeNumeric")],
                                       nDays =log(1)),type ="link",
                      se.fit =TRUE)
deathGamPredMat =do.call(cbind,
                        deathGamPred)%*%+Pmisc::ciMat()
```

	est	se
(Intercept)	5.456	0.001
cos12	0.085	0.001
sin12	0.017	0.001
cos6	-0.008	0.001
sin6	-0.003	0.001



longer time span

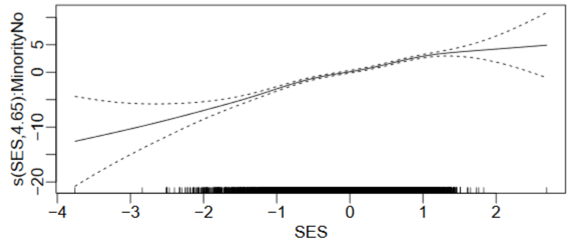


2.8 Number of Knots

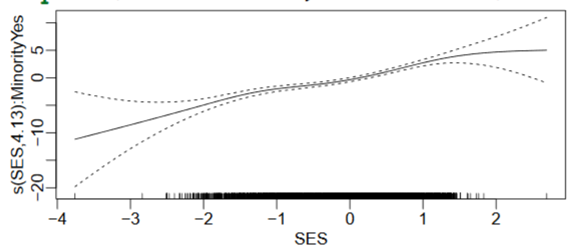
```
> mathGamIntC = gam(MathAch ~
+   s(SES, by=Minority, k=10, id=1) +
+     Minority*Sex,
+   data=MathAchieve)
> mathGamIntC$sp
s(SES):MinorityNo
        0.7492505
```

	Estimate	Std. Error
(Intercept)	12.8	0.1
MinorityYes	-2.9	0.2
SexMale	1.4	0.2
MinorityYes:SexMale	-0.1	0.3

```
> plot(mathGamIntC, select = 1)
```

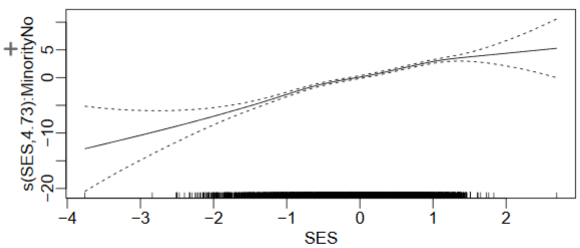


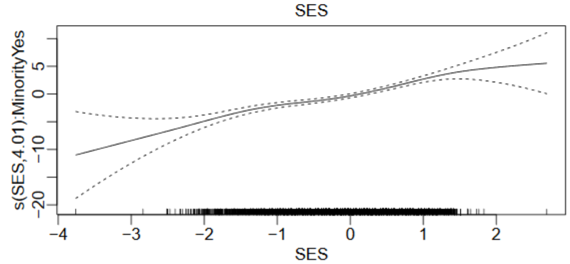

```
> plot(mathGamIntC, select = 2)
```




```
> mathGamIntC = gam(MathAch ~
+   s(SES, by=Minority, k=150, id=1) +
+     Minority*Sex,
+   data=MathAchieve)
> mathGamIntC$sp
s(SES):MinorityNo
        5105.77
```

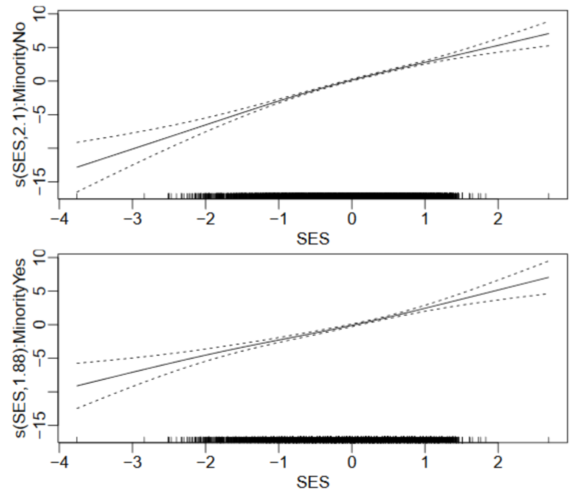
	Estimate	Std. Error
(Intercept)	12.8	0.1
MinorityYes	-2.9	0.2
SexMale	1.4	0.2
MinorityYes:SexMale	-0.1	0.3





```
> mathGamIntC = gam(MathAch ~
+   s(SES, by=Minority, k=5, id=1) +
+     Minority*Sex,
+   data=MathAchieve)
> mathGamIntC$sp
s(SES):MinorityNo
         4.511766
```

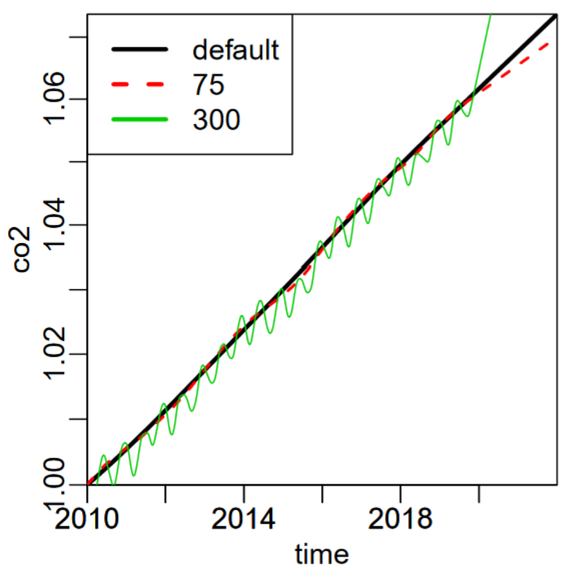
	Estimate	Std. Error
(Intercept)	12.8	0.1
MinorityYes	-2.9	0.2
SexMale	1.4	0.2
MinorityYes:SexMale	-0.1	0.3



- The more knots there are, the better approximation the model is;
- But we don't want that much preciseness/overfitting...
- If $\hat{f}$ is smooth, we don't need to many knots.
- GAM + GCV is fast, but you have to use enough basis functions. (The default number of basis functions is fairly small)

CO2 GAM

```
> res1 = mgcv::gam(logCo2 ~
+   sin12 + cos12 + sin6 + cos6 +
+   s(timeNumeric, pc=0, k=300),
+   data=co2s)
> res2 = mgcv::gam(logCo2 ~
+   sin12 + cos12 + sin6 + cos6 +
+   s(timeNumeric, pc=0,k=75),
+   data=co2s)
> resDefault = mgcv::gam(logCo2 ~
+   sin12 + cos12 + sin6 + cos6 +
+   s(timeNumeric, pc=0), data=co2s)
```



2.8.1 Regression splines

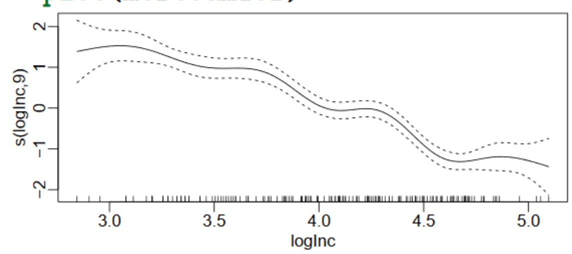
At each subset of the data between 2 knots, do a regression...

Regression splines

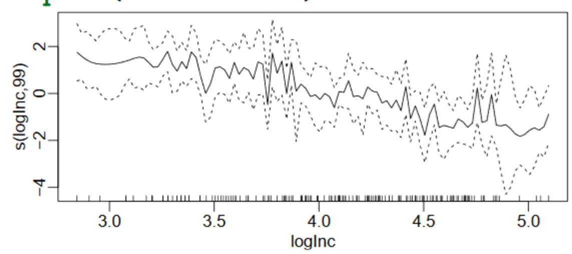
- what's faster than GCV?
- don't apply a roughness penalty
- control smoothness with the number of basis functions
- more knots means rougher $\hat{f}$
- choose knots in some ad-hoc way
- useful for models where GCV isn't possible

```
> mortGam10f = gam(logMort ~
+   s(logInc, k=10, fx=TRUE),
+   data=iMort)
> mortGam100f = gam(logMort ~
+   s(logInc, k=100, fx=TRUE),
+     data=iMort)
```

```
> plot(mortGam10f)
```




```
> plot(mortGam100f)
```



2.9 ML/model-based smoothing

Use random effects instead of penalized likelihood.

2.9.1 Random Effects

An asside: Random Walks

- RW(0), independent

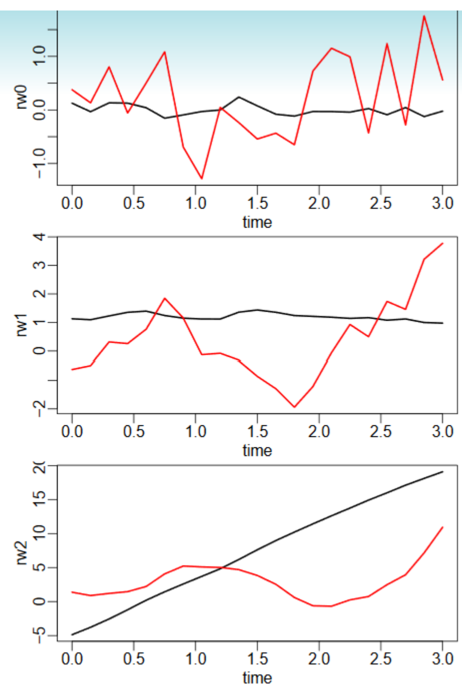
$$V_t \sim \text{iid } \mathrm{N}(0, \tau^2)$$

- RW(1), Brownian motion

$$V_{t+1}|V_k, k < t \sim \mathrm{N}(V_t, \tau^2)$$
$$V_{t+1} - V_t \sim \mathrm{N}(0, \tau^2)$$

- RW(2), Random slope

$$V_{t+1}|V_k, k < t \sim \mathrm{N}(-2V_t + V_{t-1}, \tau^2)$$
$$(V_{t+1} - V_t) - (V_t - V_{t-1}) \sim \mathrm{N}(0, \tau^2)$$
$$V_{t+1} - 2V_t + V_{t-1} \sim \mathrm{N}(0, \tau^2)$$



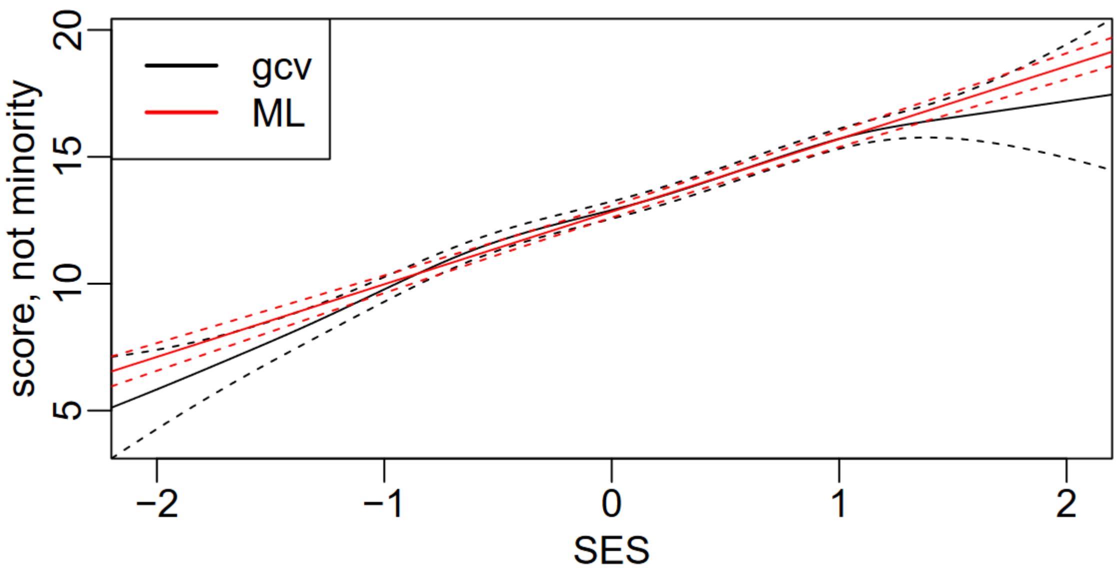
Using model-based GAM, we can apply the usual statistical machinery:

- MLE;

- conditional distribution;
- LR test

2.9.2 ML vs GCV

```
mathGamMl =gam(MathAch~s(SES,by =Minority,
                          id =1,k =30)
               + Minority*Sex,
               data =MathAchieve,
               method ="ML")
mathGamMl$sp
s(SES):MinorityNo
145931.6
```



2.9.3 LR test

Test whether the model above is simply a linear model.

```
mathLm =gam(MathAch~SES*Minority+Minority*Sex,
            data=MathAchieve)

logLik(mathLm,REML =FALSE)
'log Lik.' -23371.28 (df=7)

logLik(mathGamMl)
'log Lik.' -23371.27 (df=7.001989)
```

```
nadiv::LRTest(logLik(mathLm,REML =FALSE),
              logLik(mathGamMl),
              boundaryCorrection =TRUE)$Pval
'log Lik.' 0.5 (df=7)
```

- p-value is pretty large, so, it is sufficient to use a simple linear model for the math data.

2.9.4 Generalized Additive Mixed Model

- GAM's are already GLMM's. So, GAMM is to add additional random effects.

```
mathGamm =gamm4::gamm4(MathAch~s(SES,k =30)+
                         Minority*Sex,
                       random =~(1|School),
                       data =MathAchieve,
                       REML =FALSE)
```

$$
\begin{aligned}
&Y_{ij} \sim N(\lambda_{ij}, \tau^2) \\
&g(\lambda_{ij}) = X_{ij}\beta + f(W_{ij}) + U_i \\
&[U_1, \cdots, U_M]^T \sim MVN(0, \sigma_1^2 I) \\
&[f(w_1) \cdots, f(w_M)]^T \sim ARIMA_{0,2,1}(\sigma_2^2, 2-\sqrt{3}) \\
&\text{Or, } f(w) \sim RW(2) \text{ with variance } \sigma_2^2
\end{aligned}
$$