State-space modelling (1)

FPA2020 Lecture 11a: TMB for state-space modelling

Toshihide Kitakado

June 17, 2020

Contents


```
library(ggplot2)
library(ggfortify)
library(TMB)
```

1 An example data set

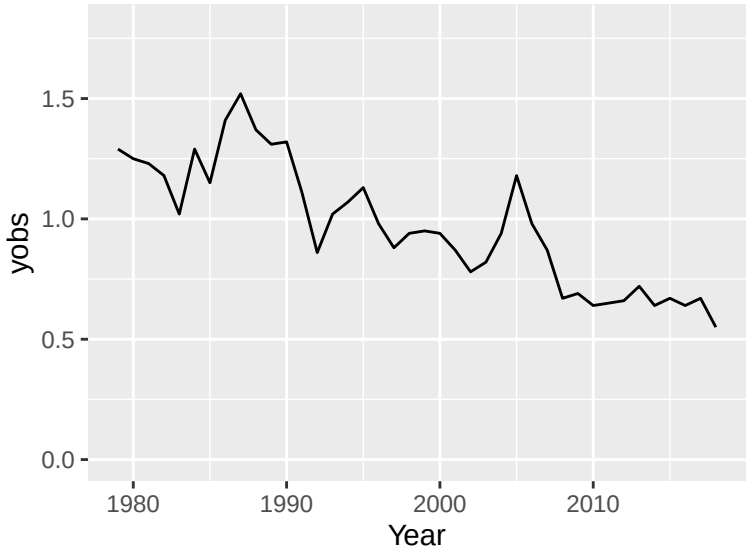
```
yobs
```

```
 [1] 1.29 1.25 1.23 1.18 1.02 1.29 1.15 1.41 1.52 1.37 1.31 1.32 1.11 0.86 1.02
[16] 1.07 1.13 0.98 0.88 0.94 0.95 0.94 0.87 0.78 0.82 0.94 1.18 0.98 0.87 0.67
[31] 0.69 0.64 0.65 0.66 0.72 0.64 0.67 0.64 0.67 0.55
```

```
Year
```

```
 [1] 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993
[16] 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008
[31] 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018
```

```
Data <- data.frame(Year,yobs)
ggplot(Data, aes(x=Year, y=yobs)) + geom_line() + ylim(0,1.8)
```




```
data <- list(TT=length(yobs), yobs=yobs)
```

2 ML State-space modelling via TMB (1) Random walk models

2.1 Definition of model

$$y_t|\mu_t \sim N(\mu_t, \sigma^2)\mu_t|\mu_{t-1} \sim N(\mu_{t-1}, \tau^2)$$

```
sink(file="RW.cpp")
cat("
#include <TMB.hpp>
template<class Type>
Type objective_function<Type>::operator() ()
{
  //Data section
  DATA_INTEGER(TT);
  //DATA_INTEGER(TF);
  DATA_VECTOR(yobs);

  //Parameter section
  PARAMETER(log_tau);
  PARAMETER(log_sigma);
  //PARAMETER(lam0);
  PARAMETER_VECTOR(mu);

  Type tau=exp(log_tau);
  Type sigma=exp(log_sigma);

  Type f=0.0;

  for(int t=1;t<TT;t++){
    f +=-dnorm(mu(t),mu(t-1),tau,true);
  }
```

```
  for(int t=0;t<TT;t++){
    f +=-dnorm(yobs(t),mu(t),sigma,true);
  }

  REPORT(mu);

  return f;
}
", fill=TRUE)
sink()
```

2.2 Compile and run

```
compile("RW.cpp")
```

```
[1] 0
```

```
dyn.load(dynlib("RW"))

parameters <- list(
  log_tau = 0.0,
  log_sigma = 0.0,
  mu0 = 0.0,
  mu=rep(0,length(yobs))
)

obj.RW <- MakeADFun(data, parameters, random="mu", DLL="RW")
```

```
Order of parameters:
[1] "log_tau"   "log_sigma" "mu0"       "mu"
Not matching template order:
[1] "log_tau"   "log_sigma" "mu"
Your parameter list has been re-ordered.
(Disable this warning with checkParameterOrder=FALSE)
```

```
fit.RW <- nlminb(obj.RW$par, obj.RW$fn, obj.RW$gr)
```

```
Optimizing tape... Done
iter: 1  value: 72.71878 mgc: 1.52 ustep: 1
iter: 2  mgc: 8.881784e-16
iter: 1  mgc: 8.881784e-16
Matching hessian patterns... Done
outer mgc:  21.59181
iter: 1  value: 17.50991 mgc: 0.2020552 ustep: 1
iter: 2  mgc: 1.526557e-15
iter: 1  value: -120.2312 mgc: 43.00125 ustep: 1
iter: 2  mgc: 1.065814e-13
iter: 1  mgc: 1.065814e-13
outer mgc:  14.75403
iter: 1  value: -120.5917 mgc: 4819.552 ustep: 1
iter: 2  mgc: 6.428066e-12
iter: 1  value: -109.2825 mgc: 13.63262 ustep: 1
```

```
iter: 2  mgc: 4.418688e-14
iter: 1  mgc: 4.418688e-14
outer mgc:  12.01488
iter: 1  value: -111.8863 mgc: 14.26983 ustep: 1
iter: 2  mgc: 5.77316e-14
iter: 1  mgc: 5.77316e-14
outer mgc:  1.958904
iter: 1  value: -120.4964 mgc: 18.52093 ustep: 1
iter: 2  mgc: 7.81597e-14
iter: 1  mgc: 7.81597e-14
outer mgc:  1.253875
iter: 1  value: -121.6624 mgc: 4.760636 ustep: 1
iter: 2  mgc: 9.237056e-14
iter: 1  mgc: 9.237056e-14
outer mgc:  0.2447954
iter: 1  value: -122.0774 mgc: 0.6376897 ustep: 1
iter: 2  mgc: 1.021405e-13
iter: 1  mgc: 1.021405e-13
outer mgc:  0.06936883
iter: 1  value: -122.4724 mgc: 0.7711513 ustep: 1
iter: 2  mgc: 8.348877e-14
iter: 1  value: -122.8832 mgc: 0.8002881 ustep: 1
iter: 2  mgc: 1.030287e-13
iter: 1  mgc: 1.030287e-13
outer mgc:  0.01636767
iter: 1  value: -123.1333 mgc: 0.5331722 ustep: 1
iter: 2  mgc: 8.704149e-14
iter: 1  mgc: 8.704149e-14
outer mgc:  0.002845523
iter: 1  value: -123.1664 mgc: 0.07117538 ustep: 1
iter: 2  mgc: 8.881784e-14
iter: 1  mgc: 8.881784e-14
outer mgc:  1.801888e-05
iter: 1  value: -123.1669 mgc: 0.001107033 ustep: 1
iter: 2  mgc: 8.526513e-14
iter: 1  mgc: 8.526513e-14
outer mgc:  3.938555e-06
iter: 1  mgc: 8.526513e-14
```

```
sdr.RW <- sdreport(obj.RW)
```

```
iter: 1  mgc: 8.526513e-14
outer mgc:  3.938555e-06
iter: 1  value: -123.1601 mgc: 0.054355 ustep: 1
iter: 2  mgc: 8.704149e-14
outer mgc:  0.05383056
iter: 1  value: -123.1737 mgc: 0.05446382 ustep: 1
iter: 2  mgc: 9.237056e-14
outer mgc:  0.05386602
iter: 1  value: -123.1338 mgc: 0.054355 ustep: 1
iter: 2  mgc: 9.414691e-14
```

```
outer mgc:  0.0104956
iter: 1  value: -123.2001 mgc: 0.05446382 ustep: 1
iter: 2  mgc: 8.704149e-14
outer mgc:  0.01048559
```

```
pl.RW <- pl <- as.list(sdr.RW, "Est"); pl.RW
```

```
$log_tau
[1] -2.181853

$log_sigma
[1] -3.25427

$mu
 [1] 1.2859318 1.2511869 1.2265791 1.1727549 1.0570531 1.2578076 1.1836190
 [8] 1.3965575 1.4946883 1.3766417 1.3153196 1.2994302 1.1078618 0.8980323
[15] 1.0130226 1.0684214 1.1103376 0.9843251 0.8952516 0.9364366 0.9471881
[22] 0.9339245 0.8687719 0.7931310 0.8296372 0.9484515 1.1394472 0.9840968
[29] 0.8637358 0.6898745 0.6857541 0.6453709 0.6508585 0.6636778 0.7079080
[36] 0.6488649 0.6655336 0.6440561 0.6572203 0.5612383

attr(,"what")
[1] "Estimate"
```

```
plsd.RW <- plsd <- as.list(sdr.RW, "Std"); plsd.RW
```

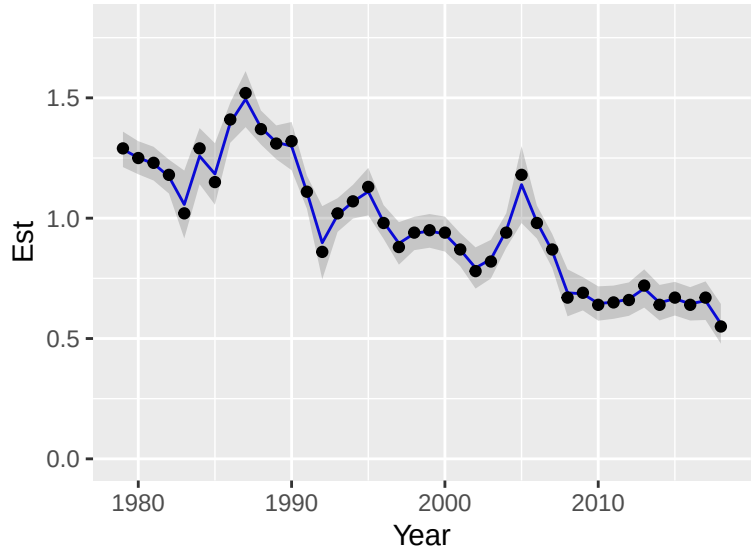
```
$log_tau
[1] 0.2285951

$log_sigma
[1] 0.9420841

$mu
 [1] 0.03761462 0.03509692 0.03576079 0.03597871 0.07190297 0.05943543
 [7] 0.06517983 0.04252215 0.05944842 0.03604628 0.03577182 0.05113690
[13] 0.03508521 0.07752609 0.03575049 0.03576204 0.05033740 0.03580595
[19] 0.04505450 0.03536144 0.03571190 0.03703342 0.03508555 0.04368932
[25] 0.04097310 0.03662958 0.08146758 0.03507438 0.03601306 0.04983947
[31] 0.03524892 0.03642228 0.03520115 0.03544300 0.04059015 0.03766852
[37] 0.03566069 0.03540655 0.04092755 0.04210136

attr(,"what")
[1] "Std. Error"
```

```
AIC.RW <- 2*fit.RW$objective + 2*3

res_df_RW <- data.frame(
  Year = Year,
  Est = pl$mu,
  SE = plsd$mu,
  LB95 = pl$mu - 1.96*plsd$mu,
  UB95 = pl$mu + 1.96*plsd$mu,
```

```
  yobs = yobs
)

dyn.unload(dynlib("RW"))
```

```
Warning: 5 external pointers will be removed
```

```
tmp_RW <- ggplot(res_df_RW, aes(Year, Est)) + geom_line(color="blue") +
  geom_ribbon(aes(ymin=LB95,ymax=UB95), alpha=0.2) + ylim(0,1.8) +
  geom_point(aes(y=yobs))
tmp_RW
```



3 ML State-space modelling via TMB (2) Smoothing model

3.1 Definition of model

$$y_t|\mu_t \sim N(\mu_t, \sigma^2) \mu_t|\mu_{t-1}, \mu_{t-2} \sim N(2\mu_{t-1} - \mu_{t-2}, \tau^2)$$

```
sink(file="Smooth.cpp")
cat("
#include <TMB.hpp>
template<class Type>
Type objective_function<Type>::operator() ()
{
  //Data section
  DATA_INTEGER(TT);
  DATA_VECTOR(yobs);

  //Parameter section
  PARAMETER(log_tau);
  PARAMETER(log_sigma);
  //PARAMETER(mu0);
  //PARAMETER(mu1);
  PARAMETER_VECTOR(mu);
```

```
  Type tau=exp(log_tau);
  Type sigma=exp(log_sigma);

  //Type small=tau;

  Type f=0.0;

  //f += -dnorm(mu(0),mu0,small,true);
  //f += -dnorm(mu(1),mu1,small,true);

  for(int t=2;t<TT;t++){
    f += -dnorm(mu(t),2*mu(t-1)-mu(t-2),tau,true);
  }

  for(int t=0;t<TT;t++){
    f+= -dnorm(yobs(t),mu(t),sigma,true);
  }

  REPORT(mu);

  return f;
}
", fill=TRUE)
sink()
```

3.2 Compile and run

```
compile("Smooth.cpp")
```

```
[1] 0
```

```
dyn.load(dynlib("Smooth"))

parameters <- list(
  log_tau = 0.0,
  log_sigma = 0.0,
  #mu0 = 0.0,
  #mu1 = 0.0,
  mu=rep(0,length(yobs))
)

obj.Smooth <- MakeADFun(data, parameters, random="mu", DLL="Smooth")

obj.Smooth$fn()
```

```
Optimizing tape... Done
iter: 1  value: 71.77885 mgc: 1.52 ustep: 1
iter: 2  mgc: 1.332268e-15
[1] 63.15701
attr(,"logarithm")
```

```
[1] TRUE
```

```
obj.Smooth$gr()
```

```
iter: 1  mgc: 1.332268e-15
Matching hessian patterns... Done
outer mgc:  23.49213
```

```
[1] 14.30458 23.49213
```

```
fit.Smooth <- nlminb(obj.Smooth$par, obj.Smooth$fn, obj.Smooth$gr)
```

```
iter: 1  mgc: 1.332268e-15
iter: 1  mgc: 1.332268e-15
outer mgc:  23.49213
iter: 1  value: 18.18553 mgc: 0.4028043 ustep: 1
iter: 2  mgc: 4.191092e-15
iter: 1  value: -118.0635 mgc: 96.06282 ustep: 1
iter: 2  mgc: 1.79412e-13
iter: 1  mgc: 1.79412e-13
outer mgc:  8.54335
iter: 1  value: -78.70091 mgc: 4.651291 ustep: 1
iter: 2  mgc: 3.108624e-14
iter: 1  value: -109.3279 mgc: 7.14052 ustep: 1
iter: 2  mgc: 9.237056e-14
iter: 1  mgc: 9.237056e-14
outer mgc:  6.391456
iter: 1  value: -111.2635 mgc: 36.68391 ustep: 1
iter: 2  mgc: 1.261213e-13
iter: 1  mgc: 1.261213e-13
outer mgc:  5.985115
iter: 1  value: -106.3904 mgc: 22.50092 ustep: 1
iter: 2  mgc: 2.038369e-13
iter: 1  mgc: 2.038369e-13
outer mgc:  4.908089
iter: 1  value: -114.7521 mgc: 19.01364 ustep: 1
iter: 2  mgc: 2.344791e-13
iter: 1  mgc: 2.344791e-13
outer mgc:  5.620371
iter: 1  value: -120.2687 mgc: 70.52932 ustep: 1
iter: 2  mgc: 5.204726e-13
iter: 1  mgc: 5.204726e-13
outer mgc:  5.401394
iter: 1  value: -117.3632 mgc: 19.85424 ustep: 1
iter: 2  mgc: 2.073897e-13
iter: 1  mgc: 2.073897e-13
outer mgc:  0.2853342
iter: 1  value: -117.2065 mgc: 0.0518034 ustep: 1
iter: 2  mgc: 3.88134e-13
iter: 1  mgc: 3.88134e-13
outer mgc:  0.05836388
iter: 1  value: -117.0669 mgc: 0.3252391 ustep: 1
```

```
iter: 2  mgc: 4.352074e-13
iter: 1  mgc: 4.352074e-13
outer mgc:  0.002813861
iter: 1  value: -117.054 mgc: 0.04835791 ustep: 1
iter: 2  mgc: 3.53495e-13
iter: 1  mgc: 3.53495e-13
outer mgc:  0.0004402757
iter: 1  value: -117.0537 mgc: 0.001961488 ustep: 1
iter: 2  mgc: 6.235013e-13
iter: 1  mgc: 6.235013e-13
outer mgc:  1.630907e-05
iter: 1  value: -117.0537 mgc: 1.088756e-05 ustep: 1
iter: 2  mgc: 3.081979e-13
iter: 1  mgc: 3.081979e-13
outer mgc:  9.199046e-08
iter: 1  mgc: 3.081979e-13
```

```
sdr.Smooth <- sdreport(obj.Smooth)
```

```
iter: 1  mgc: 3.081979e-13
outer mgc:  9.199046e-08
iter: 1  value: -117.0277 mgc: 0.05409633 ustep: 1
iter: 2  mgc: 4.192202e-13
outer mgc:  0.0137142
iter: 1  value: -117.0798 mgc: 0.05420463 ustep: 1
iter: 2  mgc: 4.884981e-13
outer mgc:  0.01372756
iter: 1  value: -117.0398 mgc: 0.05409633 ustep: 1
iter: 2  mgc: 4.805045e-13
outer mgc:  0.03838395
iter: 1  value: -117.0677 mgc: 0.05420463 ustep: 1
iter: 2  mgc: 4.192202e-13
outer mgc:  0.03841804
```

```
pl.Smooth <- pl <- as.list(sdr.Smooth, "Est"); pl.Smooth
```

```
$log_tau
[1] -2.822524

$log_sigma
[1] -2.511876

$mu
 [1] 1.2875068 1.2429228 1.1996783 1.1629149 1.1540644 1.2037376 1.2705198
 [8] 1.3593402 1.4103792 1.3910339 1.3275951 1.2350531 1.1189452 1.0304465
[15] 1.0159259 1.0301808 1.0301969 0.9943530 0.9546469 0.9353650 0.9206902
[22] 0.8972953 0.8675996 0.8569657 0.8920456 0.9581419 1.0018505 0.9600211
[29] 0.8652133 0.7607205 0.6924079 0.6574011 0.6515321 0.6612843 0.6723181
[36] 0.6696036 0.6637281 0.6493744 0.6245949 0.5824056

attr(,"what")
[1] "Estimate"
```

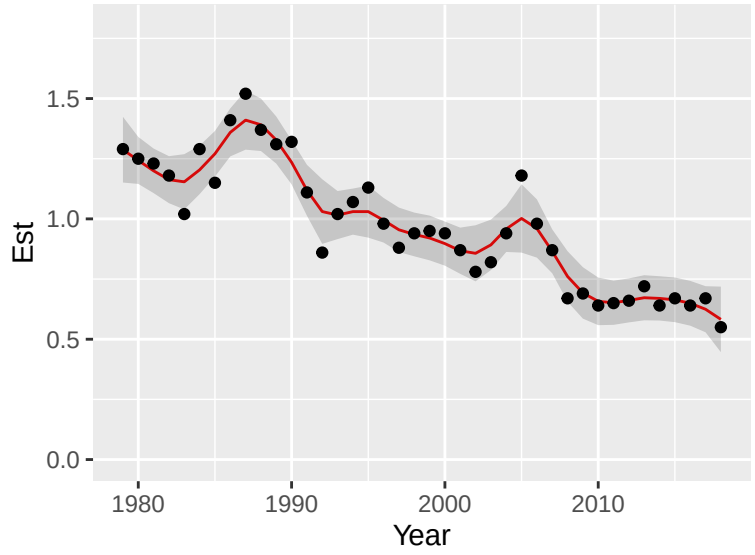
```
plsd.Smooth <- plsd <- as.list(sdr.Smooth, "Std"); plsd.Smooth
```

```
$log_tau
[1] 0.4362408

$log_sigma
[1] 0.2243587

$mu
 [1] 0.06966040 0.04981440 0.04723869 0.04981387 0.05858662 0.05143309
 [7] 0.04836167 0.05089506 0.06261757 0.05567259 0.04990499 0.04670632
[13] 0.05398086 0.06849389 0.05096460 0.04888551 0.05516785 0.04728116
[19] 0.04709054 0.04633414 0.04741409 0.04678283 0.04902222 0.05948738
[25] 0.05352145 0.04878424 0.07243226 0.06176761 0.04658154 0.05357819
[31] 0.05452457 0.05052036 0.04682091 0.04660101 0.04781171 0.04697135
[37] 0.04734257 0.04752294 0.04886326 0.06949086

attr(,"what")
[1] "Std. Error"
```

```
AIC.Smooth <- 2*fit.Smooth$objective + 2*4

res_df_Smooth <- data.frame(
  Year = Year,
  Est = pl$mu,
  SE = plsd$mu,
  LB95 = pl$mu - 1.96*plsd$mu,
  UB95 = pl$mu + 1.96*plsd$mu,
  yobs = yobs
)

dyn.unload(dynlib("Smooth"))
```

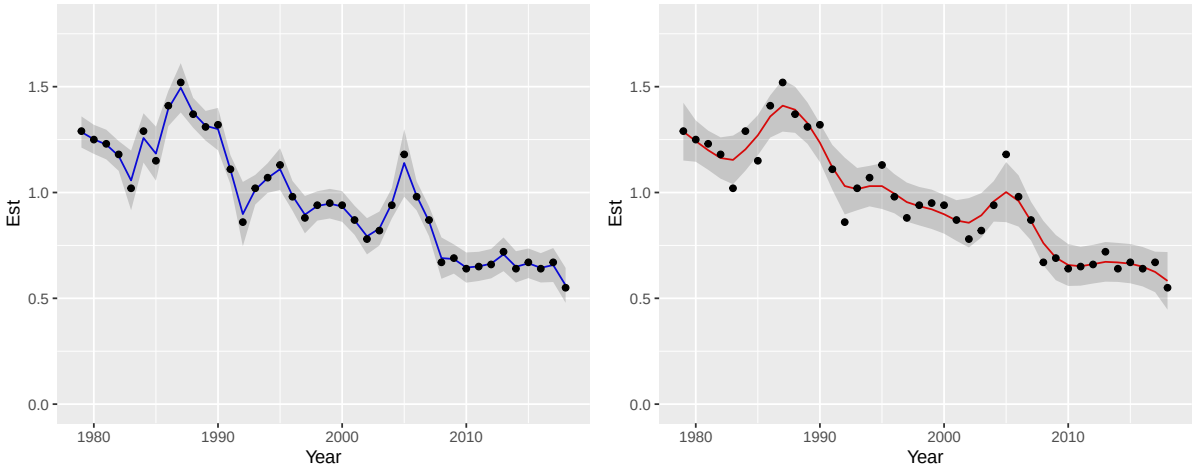
```
Warning: 6 external pointers will be removed
```

```
tmp_Smooth <- ggplot(res_df_Smooth, aes(Year, Est)) + geom_line(color="red") +
  geom_ribbon(aes(ymin=LB95,ymax=UB95), alpha=0.2) + ylim(0,1.8) +
  geom_point(aes(y=yobs))
tmp_Smooth
```



4 Comparison of models between RW and Smooth

```
gridExtra::grid.arrange(tmp_RW, tmp_Smooth,nrow=1)
```




```
data.frame(Model=c("RW","Smooth"), AIC=c(AIC.RW, AIC.Smooth))
```

```
   Model       AIC
1     RW -45.64828
2 Smooth -26.94399
```

5 Random walk in the presense of missing data

5.1 Mising data

I intentionally remove some of data to make mising values in the data set. The latent model can easily deal with this situation.

```
yobs_miss <- yobs
yobs_miss[c(7:10, 26:27)] <- NA
yobs_miss
```

```
 [1] 1.29 1.25 1.23 1.18 1.02 1.29   NA   NA   NA   NA 1.31 1.32 1.11 0.86 1.02
[16] 1.07 1.13 0.98 0.88 0.94 0.95 0.94 0.87 0.78 0.82   NA   NA 0.98 0.87 0.67
[31] 0.69 0.64 0.65 0.66 0.72 0.64 0.67 0.64 0.67 0.55
```

```
data_miss <- list(TT=length(yobs_miss), yobs=yobs_miss)
```

```
sink(file="RWmiss.cpp")
cat("
#include <TMB.hpp>

template<class Type>
bool isNA(Type x){
  return R_IsNA(asDouble(x));
}

template<class Type>
Type objective_function<Type>::operator() ()
{
  //Data section
  DATA_INTEGER(TT);
  //DATA_INTEGER(TF);
  DATA_VECTOR(yobs);
  //int timeSteps=yobs.size();

  //Parameter section
  PARAMETER(log_tau);
  PARAMETER(log_sigma);
  //PARAMETER(mu0);
  PARAMETER_VECTOR(mu);

  Type tau=exp(log_tau);
  Type sigma=exp(log_sigma);

  //Type small=tau;

  Type f=0.0;
  //Type f +=-dnorm(mu(0),mu0,small,true);

  for(int t=1;t<TT;t++){
    f +=-dnorm(mu(t),mu(t-1),tau,true);
  }

  for(int t=0;t<TT;t++){
    if(!isNA(yobs(t))){
      f +=-dnorm(yobs(t),mu(t),sigma,true);
    }
  }

  REPORT(mu);

  return f;
```

```
}
", fill=TRUE)
sink()
```

5.2 Compile and run

```
compile("RWmiss.cpp")
```

```
[1] 0
```

```
dyn.load(dynlib("RWmiss"))

parameters <- list(
  log_tau = 0.0,
  log_sigma = 0.0,
  #mu0 = 0.0,
  mu=rep(0,length(yobs))
)

obj.RWmiss <- obj <- MakeADFun(data_miss, parameters, random="mu", DLL="RWmiss")
fit.RWmiss <- nlminb(obj$par, obj$fn, obj$gr)
```

```
Optimizing tape... Done
iter: 1  value: 67.1587 mgc: 1.32 ustep: 1
iter: 2  mgc: 8.881784e-16
iter: 1  mgc: 8.881784e-16
Matching hessian patterns... Done
outer mgc:  17.87004
iter: 1  value: 16.28447 mgc: 0.1415626 ustep: 1
iter: 2  mgc: 1.179612e-15
iter: 1  value: -117.353 mgc: 30.34135 ustep: 1
iter: 2  mgc: 6.483702e-14
iter: 1  value: 13740.78 mgc: 48285.65 ustep: 1
iter: 2  mgc: 1.181206e-10
iter: 1  mgc: 6.483702e-14
outer mgc:  4.844637
iter: 1  value: -94.40448 mgc: 210.7243 ustep: 1
iter: 2  mgc: 2.674805e-13
iter: 1  value: -112.6263 mgc: 6.120432 ustep: 1
iter: 2  mgc: 7.727152e-14
iter: 1  mgc: 7.727152e-14
outer mgc:  2.085496
iter: 1  value: -116.8065 mgc: 13.4286 ustep: 1
iter: 2  mgc: 7.638334e-14
iter: 1  mgc: 7.638334e-14
outer mgc:  0.6934113
iter: 1  value: -117.2067 mgc: 2.525797 ustep: 1
iter: 2  mgc: 6.261658e-14
iter: 1  mgc: 6.261658e-14
outer mgc:  0.5212379
iter: 1  value: -117.6643 mgc: 3.254497 ustep: 1
```

```
iter: 2  mgc: 8.526513e-14
iter: 1  mgc: 8.526513e-14
outer mgc:  0.1516149
iter: 1  value: -118.976 mgc: 6.238648 ustep: 1
iter: 2  mgc: 1.150191e-13
iter: 1  mgc: 1.150191e-13
outer mgc:  0.1548811
iter: 1  value: -119.3319 mgc: 1.064431 ustep: 1
iter: 2  mgc: 1.101341e-13
iter: 1  mgc: 1.101341e-13
outer mgc:  0.07135506
iter: 1  value: -119.6797 mgc: 1.104477 ustep: 1
iter: 2  mgc: 8.881784e-14
iter: 1  mgc: 8.881784e-14
outer mgc:  0.01614018
iter: 1  value: -119.8105 mgc: 0.4252094 ustep: 1
iter: 2  mgc: 1.030287e-13
iter: 1  mgc: 1.030287e-13
outer mgc:  0.001186114
iter: 1  value: -119.8219 mgc: 0.03753266 ustep: 1
iter: 2  mgc: 1.030287e-13
iter: 1  mgc: 1.030287e-13
outer mgc:  3.697876e-05
iter: 1  value: -119.8222 mgc: 0.0008459033 ustep: 1
iter: 2  mgc: 1.119105e-13
iter: 1  mgc: 1.119105e-13
outer mgc:  1.320863e-06
iter: 1  mgc: 1.119105e-13
```

```
sdr.RWmiss <- sdr <- sdreport(obj)
```

```
iter: 1  mgc: 1.119105e-13
outer mgc:  1.320863e-06
iter: 1  value: -119.8093 mgc: 0.06750117 ustep: 1
iter: 2  mgc: 8.21565e-14
outer mgc:  0.04172973
iter: 1  value: -119.8351 mgc: 0.0676363 ustep: 1
iter: 2  mgc: 9.903189e-14
outer mgc:  0.04175461
iter: 1  value: -119.7951 mgc: 0.06750117 ustep: 1
iter: 2  mgc: 8.704149e-14
outer mgc:  0.01050477
iter: 1  value: -119.8493 mgc: 0.0676363 ustep: 1
iter: 2  mgc: 8.304468e-14
outer mgc:  0.01050611
```

```
pl.RWmiss <- pl <- as.list(sdr, "Est"); pl
```

```
$log_tau
[1] -2.354248

$log_sigma
```

```
[1] -3.28203

$mu
 [1] 1.2847582 1.2512356 1.2256150 1.1719511 1.0668119 1.2610491 1.2701366
 [8] 1.2792240 1.2883115 1.2973989 1.3064864 1.2931031 1.1077063 0.9076409
[15] 1.0122537 1.0673262 1.1052989 0.9853003 0.8991987 0.9358786 0.9462011
[22] 0.9322286 0.8685555 0.7956445 0.8227846 0.8677334 0.9126821 0.9576309
[29] 0.8595223 0.6944060 0.6853735 0.6467532 0.6513214 0.6643401 0.7051150
[36] 0.6506960 0.6646812 0.6446513 0.6543679 0.5641127

attr(,"what")
[1] "Estimate"
```

```
plsd.RWmiss <- plsd <- as.list(sdr, "Std"); plsd
```

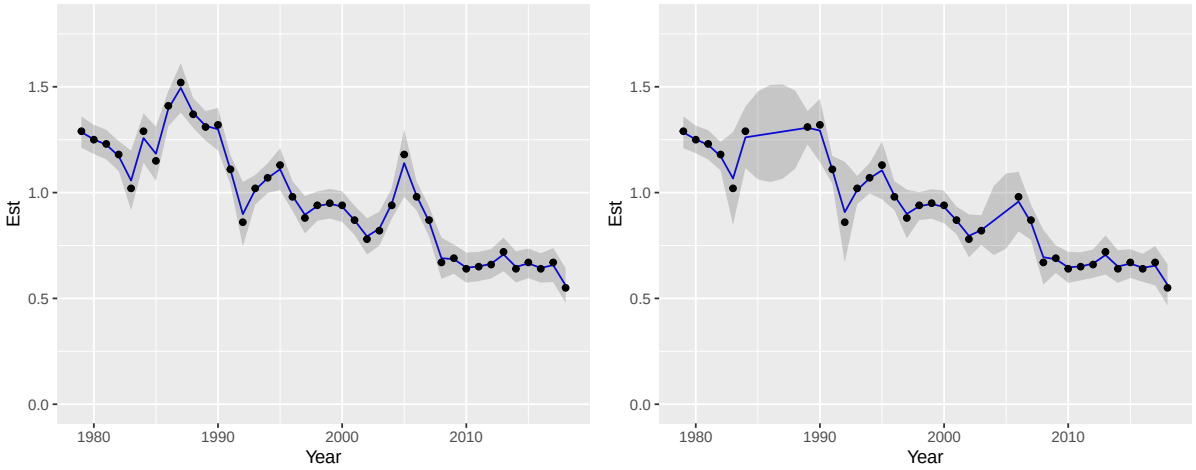
```
$log_tau
[1] 0.3591674

$log_sigma
[1] 1.287797

$mu
 [1] 0.03815362 0.03328694 0.03530647 0.03434234 0.11174564 0.07409240
 [7] 0.10596988 0.11710169 0.11375086 0.09438733 0.04027328 0.07640492
[13] 0.03325777 0.12135746 0.03397863 0.03637378 0.06976335 0.03518699
[19] 0.05841708 0.03378431 0.03555043 0.03924051 0.03328408 0.05173547
[25] 0.03584911 0.08308243 0.09050749 0.07177679 0.04259522 0.06576198
[31] 0.03336055 0.03738934 0.03382777 0.03408768 0.04706386 0.03946629
[37] 0.03468287 0.03384832 0.04733212 0.04962866

attr(,"what")
[1] "Std. Error"
```

```
res_df_RWmiss <- data.frame(
  Year = Year,
  Est = pl$mu,
  SE = plsd$mu,
  LB95 = pl$mu - 1.96*plsd$mu,
  UB95 = pl$mu + 1.96*plsd$mu,
  yobs = yobs_miss
)
dyn.unload(dynlib("RWmiss"))
```

```
Warning: 6 external pointers will be removed
```

5.3 Results with and without missing data

```
tmp_RWmiss <- ggplot(res_df_RWmiss, aes(Year, Est)) + geom_line(color="blue") +
  geom_ribbon(aes(ymin=LB95,ymax=UB95), alpha=0.2) + ylim(0,1.8) +
  geom_point(aes(y=yobs))

gridExtra::grid.arrange(tmp_RW, tmp_RWmiss,nrow=1)
```


Est
1.5
1.0
0.5
0.0
1980
1990
2000
2010
Year
Est
1.5
1.0
0.5
0.0
1980
1990
2000
2010
Year