

Sharing data between R and ADMB

December 11-12, 2013.

Getting data from admb

- write text files from admb
 - ▶ fine small number of files/variables
 - ▶ verbose
 - ▶ violates DRY

in tpl:

```
ofstream maturity("maturity.txt",ios::out);
maturity << "year  age  mat " << endl;
maturity.setf(ios::fixed);
maturity.setf(ios::right);
for (i=fyear;i<=lyear;i++)
  for (j=fage;j<=lage;j++)
    maturity << i << j << mat(i,j) << endl;
```

Getting data from admb

- parse admb report file

Getting data from admb

- parse admb report file
 - ▶ may conflict with other tools

Getting data from admb

- parse admb report file
 - ▶ may conflict with other tools
 - ▶ fragile

Getting data from admb

- parse admb report file
 - ▶ may conflict with other tools
 - ▶ fragile
- Don't do it.

- Martin, Jennifer L. and Prager, Michael H. and Stephens, Andi (eds.) (2009) User's Guide to ADMB2R: A Set of AD Model Builder Output Routines Compatible with the R Statistics Language. Miami, FL, NOAA/National Marine Fisheries Service/Southeast Fisheries Science Center, (NOAA Technical Memorandum NMFS-SEFSC, 546)

- Martin, Jennifer L. and Prager, Michael H. and Stephens, Andi (eds.) (2009) User's Guide to ADMB2R: A Set of AD Model Builder Output Routines Compatible with the R Statistics Language. Miami, FL, NOAA/National Marine Fisheries Service/Southeast Fisheries Science Center, (NOAA Technical Memorandum NMFS-SEFSC, 546)
- available at: <http://code.google.com/p/admb-project/downloads/list>

admb2R

- my preferred approach
 - ▶ place admb2r.cpp on PATH
 - ▶ create *.cxx file with desired elements
 - ▶ add 3 statements to tpl
 - ▶ creates an *.rdat file that can be read directly into R using `dget()`

tpl changes:

```
GLOBALS_SECTION
  #include "admodel.h"
  #include "admb2r.cpp"
...
REPORT_SECTION
  #include "make-rdat.cxx"  // for ADMB2R
```

admb2R- basic *.cxx syntax

- see manual for complete details

open and close file:

```
open_r_file(adprogram_name + ".rdat", 6, -999);  
...  
close_r_file(
```

info_list:

```
open_r_info_list("info", true);  
  wrt_r_item("model", (char*)(adprogram_name));  
  wrt_r_item("species", "LakeWhitefish");  
  wrt_r_item("units.len", "mm");  
close_r_info_list();
```

admb2R - scalars

- written into vectors, lists or info_lists
- logical groupings - dims, brps, etc
- `wrt_r_item("R_name", <admb_name>);`

scalars:

```
open_r_info_list("dims", false);  
  wrt_r_item("fyear", fyear);  
  wrt_r_item("lyear", lyear);  
  wrt_r_item("fage", fage);  
  wrt_r_item("lage", lage);  
  wrt_r_item("sp_time", sp_time);  
close_r_info_list();
```

admb2R - vectors

vector - element by element:

```
open_r_vector(name)
  wrt_r_item(name, value)
  ...
close_r_vector()
```

- OR:

vector - all at once:

```
wrt_r_complete_vector("<R name>", <admb name>, <name v...>
wrt_r_complete_vector("Biomass", BIOMASS, years);
```

admb2R - matrices

- optional arguments to specify row and column names
 - ▶ we used matrix indices in this example
- options for missing values

matrices

```
open_r_matrix("<R name>");  
  wrt_r_matrix(<admb name>,1,1);  
close_r_matrix();  
  
open_r_matrix("GLpa");  
  wrt_r_matrix(PAG,1,1);  
close_r_matrix();
```

admb2R - data frames

- creates an R data frame
- allows different data types in each column
- extremely flexible - column and row names, ragged arrays, missing values

data frames

```
open_r_df(name, start, stop, writerow)
  wrt_r_df_col(name, xx, shift, isna, na_vector)
  wrt_r_df_col(name, start, stop, inc, isna, na_vector)
  ...
  wrt_r_namevector(rowvec, i_start, i_stop)
  wrt_r_namevector(start, stop, inc)
close_r_df()
```

ADMButils

- collection of R-functions for working with admb:

ADMButils

- collection of R-functions for working with admb:
 - ▶ `read.par()`

ADMButils

- collection of R-functions for working with admb:
 - ▶ `read.par()`
 - ▶ `read.std()`

ADMButils

- collection of R-functions for working with admb:
 - ▶ `read.par()`
 - ▶ `read.std()`
 - ▶ `read.fit()`

ADMButils

- collection of R-functions for working with admb:
 - ▶ `read.par()`
 - ▶ `read.std()`
 - ▶ `read.fit()`
 - ▶ `readcxx()`

ADMButils

- collection of R-functions for working with admb:
 - ▶ `read.par()`
 - ▶ `read.std()`
 - ▶ `read.fit()`
 - ▶ `readcxx()`
 - ▶ `readmcmc()`

ADMButils

- collection of R-functions for working with admb:
 - ▶ `read.par()`
 - ▶ `read.std()`
 - ▶ `read.fit()`
 - ▶ `readcxx()`
 - ▶ `readmcmc()`
 - ▶ `write.pin()`

ADMButils

- collection of R-functions for working with admb:

- ▶ `read.par()`
- ▶ `read.std()`
- ▶ `read.fit()`
- ▶ `readcxx()`
- ▶ `readmcmc()`
- ▶ `write.pin()`
- ▶ `write.dat()`

ADMButils

- collection of R-functions for working with admb:

- ▶ `read.par()`
- ▶ `read.std()`
- ▶ `read.fit()`
- ▶ `readcxx()`
- ▶ `readmcmc()`
- ▶ `write.pin()`
- ▶ `write.dat()`

- source code is available at
<https://github.com/AdamCottrill/ADMButils>

ADMButils

- collection of R-functions for working with admb:
 - ▶ `read.par()`
 - ▶ `read.std()`
 - ▶ `read.fit()`
 - ▶ `readcxx()`
 - ▶ `readmcmc()`
 - ▶ `write.pin()`
 - ▶ `write.dat()`
- source code is available at <https://github.com/AdamCottrill/ADMButils>
- other options exist - R2admb, admbGLMM might be worth exploring

read.par()

- return contents of par file as a named list including gradient, parameter count and objective function
- if reduced==TRUE, parameter estimates omitted

```
read.par()
```

```
fit <- read.par("Von_Bert", reduce=FALSE)
str(fit)
fit$par.cnt
fit$obj.fct
fit$gradient
fit$Linf
```

read.std()

- return named list of values in std file produced by admb
- values are returned as a data frame or a named list
- names of list objects are “foo” and “foo.se”

read.std()

```
fit <- read.std("Von_Bert")  
fit  
str(fit)
```

```
fit <- read.std("Von_Bert", as.df=FALSE)  
fit  
str(fit)
```

`read.fit()`

- a convenience wrapper calls:
 - ▶ `read.par()`
 - ▶ `read.std()`
- returns named list that includes contents of `par`, `std`, `cov` and `cor` files

`read.fit()`

```
fit <- read.fit("Von_Bert")
str(fit)
fit$est
fit$std
fit$cor
fit$cov
```

readcxx()

- a convenience wrapper calls:
 - ▶ `read.par()`
 - ▶ `read.std()`
 - ▶ `dget()`
- for use with `*.rdat` files created by `admb2r`
- returns named list that includes contents of `rdat`, `par`, `std`, `cov` and `cor` files

readcxx()

```
fit <- readcxx("Von_Bert")  
str(fit)  
fit$std
```

readmcmc()

- reads in results of mcmc simulations from admb
- returns an coda mcmc object
- default input is a csv file with header
- numerous options to accomodate legacy approaches

readmcmc()

```
mcmc <- read.mcmc(mcmc.file="mcmc.csv")  
class(mcmc)  
str(mcmc)  
plot(mcmc)  
xyplot(mcmc, aspect="fill", layout=c(2,2))
```

write.dat()

- a helper function for writing admb *.dat files
- takes a named list of elements
- factors are convert to numeric values and key written into file header

write.dat()

```
fname <- "Von_Bert.dat"
data_list <- list(Linf_L=300, Linf_U=1200,
                  nobs=nrow(mydata),
                  data=mydata)
write.dat(L=data_list, name=fname)
```

write.pin()

- a helper function for writing admb pin files
- takes a named list of elements

```
write.pin()
```

```
fname <- "Von_Bert.pin"  
pin_list <- list(Linf=900, k=0.3, t0=0.0)  
write.pin(L=pin_list, name=fname)
```

calling admb from R

- possible to use R to call admb executable

call an admb executable:

```
shell.exec("Von_Bert.exe")
```

- easy simulations:
 - ▶ write dat and pin file
 - ▶ call admb executable
 - ▶ read results and analyse in R
 - ▶ copy results to archive directory
 - ▶ repeat