RSiena: Workshop October 2010

R.M. Ripley

Department of Statistics
University of Oxford

2010

Outline

Introductory remarks	I am experiencing a little sense of déjà vu
Debugging	A short demo based on two recent errors encountered by some of you.
New Effects	Brief outline of how to create a new effect.
Practical session	You make changes of your choice to RSiena. I help!

Debugging

What to do if you encounter an error:

1. Read the manual, particularly the script.
2. Read the help pages.
3. Google: if the problem has arisen before there may be a message on RSiena-help or the stocnet yahoo user group about it.
4. Try to debug it yourself
5. Finally, if all else fails, email one of the help lists:
   rsiena-help@lists.r-forge.r-project.org
   or
   the Stocnet yahoo user group

Debugging

What to do if you encounter an error:

1. Read the manual, particularly the script.
2. Read the help pages.
3. Google: if the problem has arisen before there may be a message on RSiena-help or the stocnet yahoo user group about it.
4. Try to debug it yourself
5. Finally, if all else fails, email one of the help lists:
   rsiena-help@lists.r-forge.r-project.org
   or
   the Stocnet yahoo user group

Debugging

What to do if you encounter an error:

1. Read the manual, particularly the script.
2. Read the help pages.
3. Google: if the problem has arisen before there may be a message on RSiena-help or the stocnet yahoo user group about it.
4. Try to debug it yourself
5. Finally, if all else fails, email one of the help lists:
   rsiena-help@lists.r-forge.r-project.org
   or
   the Stocnet yahoo user group

Debugging

What to do if you encounter an error:

1. Read the manual, particularly the script.
2. Read the help pages.
3. Google: if the problem has arisen before there may be a message on RSiena-help or the stocnet yahoo user group about it.
4. Try to debug it yourself
5. Finally, if all else fails, email one of the help lists:
   rsiena-help@lists.r-forge.r-project.org
   or
   the Stocnet yahoo user group

Debugging

What to do if you encounter an error:

1. Read the manual, particularly the script.
2. Read the help pages.
3. Google: if the problem has arisen before there may be a message on RSiena-help or the stocnet yahoo user group about it.
4. Try to debug it yourself
5. Finally, if all else fails, email one of the help lists:
   rsiena-help@lists.r-forge.r-project.org
   or
   the Stocnet yahoo user group

Simple R error (1)

```
library(RSiena)
bfriendship <- sienaNet(array(rbinom(50*10*3,1,.1),
    dim=c(50, 10, 3)), "bipartite",
    nodeSet=list("senders", "receivers"))
senders <- sienaNodeSet(50, nodeSetName="senders")
receivers <- sienaNodeSet(10, nodeSetName="receivers")
mydata <- sienaDataCreate(bfriendship,
    nodeSets=list(senders,receivers))
myeff <- getEffects(mydata)
```

```
Error in attr(xx$depvars[[j]], "nodeSet")[1] ==
  nodeSets[1] :
comparison of these types is not implemented
```

Solution: read the manual carefully:

Compare the sample script:

```
bfriendship <- sienaNet(array(c(friend.data.w1,
friend.data.w2, friend.data.w3), dim=c(50, 50, 3)),
"bipartite", nodeSet=c("senders", "receivers"))
```

with your version:

```
bfriendship <- sienaNet(array(rbinom(50*10*3,1,.1),
    dim=c(50, 10, 3)), "bipartite",
    nodeSet=list("senders", "receivers"))
```

And correct yours.

Compare the sample script:

```
bfriendship <- sienaNet(array(c(friend.data.w1,
friend.data.w2, friend.data.w3), dim=c(50, 50, 3)),
"bipartite", nodeSet=c("senders", "receivers"))
```

with your version:

```
bfriendship <- sienaNet(array(rbinom(50*10*3,1,.1),
    dim=c(50, 10, 3)), "bipartite",
    nodeSet=list("senders", "receivers"))
```

And correct yours.

Another simple R error

```
library(RSiena)
mymodel <- sienaModelCreate(nsub=2, n3=100)
bfriendship <- sienaNet(array(c(s501,
    s502, s503), dim=c(50, 50, 3)), "bipartite",
    nodeSet=c("senders", "receivers"))
senders <- sienaNodeSet(50, nodeSetName="senders")
receivers <- sienaNodeSet(50, nodeSetName="receivers")
mydata <- sienaDataCreate(bfriendship,
    nodeSets=list(senders,receivers))
myeff <- getEffects(mydata)
ans <- siena07(mymodel, data=mydata, effects=myeff)
tt <- sienaTimeTest(ans)
```

```
Error in if (dim(sienaFit$sf2[, , -escreen])[3] ==
 dim(sienaFit$effects[,: argument is of length zero
```

Error=recover

Type

```
options(error=recover)
```

and rerun:

```
tt <- sienaTimeTest(ans)
```

This gives:

```
Error in if (dim(sienaFit$sf2[, , -escreen])[3] ==
  dim(sienaFit$effects[,  :
  argument is of length zero

Enter a frame number, or 0 to exit

1: sienaTimeTest(ans)
```

Error=recover

Type

```
options(error=recover)
```

and rerun:

```
tt <- sienaTimeTest(ans)
```

This gives:

```
Error in if (dim(sienaFit$sf2[, , -escreen])[3] ==
  dim(sienaFit$effects[, :
  argument is of length zero

Enter a frame number, or 0 to exit

1: sienaTimeTest(ans)
```

Error=recover

Type

```
options(error=recover)
```

and rerun:

```
tt <- sienaTimeTest(ans)
```

This gives:

```
Error in if (dim(sienaFit$sf2[, , -escreen])[3] ==
  dim(sienaFit$effects[,  :
  argument is of length zero

Enter a frame number, or 0 to exit

1: sienaTimeTest(ans)
```

Using recover

- Type 1 to inspect the frame of the function with the problem. (If more than one level, usually start at the bottom.)
- Examine the variables that are involved by typing their names:

```
escreen
[1] -99999
dim(sienaFit$sf2)
[1] 100   2   1
dim(sienaFit$sf2[ , , -escreen])
[1] 100   2
dim(sienaFit$sf2[ , , -escreen])[3]
[1] NA
```

- Why? By default, extraction removes dimensions of length 1.
- Solution? drop=FALSE

```
dim(sienaFit$sf2[ , , -escreen, drop=FALSE])
[1] 100   2   1
dim(sienaFit$sf2[ , , -escreen, drop=FALSE])[3]
[1] 1
```

- By default, bipartite networks only have one effect included, hence this error tends to arise for bipartite networks.

Using recover

- Type 1 to inspect the frame of the function with the problem. (If more than one level, usually start at the bottom.)
- Examine the variables that are involved by typing their names:

```
escreen
-99999
```

```
dim(sienaFit$sf2)
100 2 1
```

```
dim(sienaFit$sf2[ , , -escreen])
100 2
```

```
dim(sienaFit$sf2[ , , -escreen])[3]
NA
```

- Why? By default, extraction removes dimensions of length 1.
- Solution? drop=FALSE

```
dim(sienaFit$sf2[ , , -escreen, drop=FALSE])
100 2 1
```

```
dim(sienaFit$sf2[ , , -escreen, drop=FALSE])[3]
1
```

- By default, bipartite networks only have one effect included, hence this error tends to arise for bipartite networks.

Using recover

- Type 1 to inspect the frame of the function with the problem. (If more than one level, usually start at the bottom.)
- Examine the variables that are involved by typing their names:

```
escreen  -99999
dim(sienaFit$sf2)  100 2 1
dim(sienaFit$sf2[ , , -escreen])  100 2
dim(sienaFit$sf2[ , , -escreen])[3]  NA
```

- Why? By default, extraction removes dimensions of length 1.
- Solution? drop=FALSE

```
dim(sienaFit$sf2[ , , -escreen, drop=FALSE])  100 2 1
dim(sienaFit$sf2[ , , -escreen, drop=FALSE])[3]  1
```

- By default, bipartite networks only have one effect included, hence this error tends to arise for bipartite networks.

Using recover

- Type 1 to inspect the frame of the function with the problem. (If more than one level, usually start at the bottom.)
- Examine the variables that are involved by typing their names:

```
escreen  -99999
dim(sienaFit$sf2)  100 2 1
```

```
dim(sienaFit$sf2[ , , -escreen])  100 2
dim(sienaFit$sf2[ , , -escreen])[3]  NA
```

- Why? By default, extraction removes dimensions of length 1.
- Solution? drop=FALSE

```
dim(sienaFit$sf2[ , , -escreen, drop=FALSE])  100 2 1
dim(sienaFit$sf2[ , , -escreen, drop=FALSE])[3]  1
```

- By default, bipartite networks only have one effect included, hence this error tends to arise for bipartite networks.

Using recover

- Type 1 to inspect the frame of the function with the problem. (If more than one level, usually start at the bottom.)
- Examine the variables that are involved by typing their names:

```
                          escreen  -99999
                  dim(sienaFit$sf2)  100 2 1
dim(sienaFit$sf2[ , , -escreen])  100 2
dim(sienaFit$sf2[ , , -escreen])[3]  NA
```

- Why? By default, extraction removes dimensions of length 1.
- Solution? drop=FALSE

```
dim(sienaFit$sf2[ , , -escreen, drop=FALSE])  100 2 1
dim(sienaFit$sf2[ , , -escreen, drop=FALSE])[3]  1
```

- By default, bipartite networks only have one effect included, hence this error tends to arise for bipartite networks.

Using recover

- Type 1 to inspect the frame of the function with the problem. (If more than one level, usually start at the bottom.)
- Examine the variables that are involved by typing their names:

```
                                  escreen  -99999
                        dim(sienaFit$sf2)  100 2 1
     dim(sienaFit$sf2[ , , -escreen])  100 2
dim(sienaFit$sf2[ , , -escreen])[3]  NA
```

- Why? By default, extraction removes dimensions of length 1.
- Solution? drop=FALSE

```
dim(sienaFit$sf2[ , , -escreen, drop=FALSE])  100 2 1
dim(sienaFit$sf2[ , , -escreen, drop=FALSE])[3]  1
```

- By default, bipartite networks only have one effect included, hence this error tends to arise for bipartite networks.

Using recover

- Type 1 to inspect the frame of the function with the problem. (If more than one level, usually start at the bottom.)
- Examine the variables that are involved by typing their names:

```
                                  escreen  -99999
                        dim(sienaFit$sf2)  100 2 1
     dim(sienaFit$sf2[ , , -escreen])  100 2
dim(sienaFit$sf2[ , , -escreen])[3]  NA
```

- Why? By default, extraction removes dimensions of length 1.
- Solution? drop=FALSE

```
     dim(sienaFit$sf2[ , , -escreen, drop=FALSE])  100 2 1
dim(sienaFit$sf2[ , , -escreen, drop=FALSE])[3]  1
```

- By default, bipartite networks only have one effect included, hence this error tends to arise for bipartite networks.

Using recover

- Type 1 to inspect the frame of the function with the problem. (If more than one level, usually start at the bottom.)
- Examine the variables that are involved by typing their names:

escreen	-99999
dim(sienaFit$sf2)	100 2 1
dim(sienaFit$sf2[, , -escreen])	100 2
dim(sienaFit$sf2[, , -escreen])[3]	NA

- Why? By default, extraction removes dimensions of length 1.
- Solution? drop=FALSE

dim(sienaFit$sf2[, , -escreen, drop=FALSE])	100 2 1
dim(sienaFit$sf2[, , -escreen, drop=FALSE])[3]	1

- By default, bipartite networks only have one effect included, hence this error tends to arise for bipartite networks.

Using recover

- Type 1 to inspect the frame of the function with the problem. (If more than one level, usually start at the bottom.)
- Examine the variables that are involved by typing their names:

escreen	-99999
dim(sienaFit$sf2)	100 2 1
dim(sienaFit$sf2[, , -escreen])	100 2
dim(sienaFit$sf2[, , -escreen])[3]	NA

- Why? By default, extraction removes dimensions of length 1.
- Solution? drop=FALSE

dim(sienaFit$sf2[, , -escreen, drop=FALSE])	100 2 1
dim(sienaFit$sf2[, , -escreen, drop=FALSE])[3]	1

- By default, bipartite networks only have one effect included, hence this error tends to arise for bipartite networks.

Using recover

- Type 1 to inspect the frame of the function with the problem. (If more than one level, usually start at the bottom.)
- Examine the variables that are involved by typing their names:

escreen	-99999
dim(sienaFit$sf2)	100 2 1
dim(sienaFit$sf2[, , -escreen])	100 2
dim(sienaFit$sf2[, , -escreen])[3]	NA

- Why? By default, extraction removes dimensions of length 1.
- Solution? drop=FALSE

dim(sienaFit$sf2[, , -escreen, drop=FALSE])	100 2 1
dim(sienaFit$sf2[, , -escreen, drop=FALSE])[3]	1

- By default, bipartite networks only have one effect included, hence this error tends to arise for bipartite networks.

Using recover

- Type 1 to inspect the frame of the function with the problem. (If more than one level, usually start at the bottom.)
- Examine the variables that are involved by typing their names:

```
                                  escreen  -99999
                        dim(sienaFit$sf2)  100 2 1
     dim(sienaFit$sf2[ , , -escreen])  100 2
dim(sienaFit$sf2[ , , -escreen])[3]  NA
```

- Why? By default, extraction removes dimensions of length 1.
- Solution? drop=FALSE

```
     dim(sienaFit$sf2[ , , -escreen, drop=FALSE])  100 2 1
dim(sienaFit$sf2[ , , -escreen, drop=FALSE])[3]  1
```

- By default, bipartite networks only have one effect included, hence this error tends to arise for bipartite networks.

Using recover

- Type 1 to inspect the frame of the function with the problem. (If more than one level, usually start at the bottom.)
- Examine the variables that are involved by typing their names:

```
                                  escreen  -99999
                        dim(sienaFit$sf2)  100 2 1
    dim(sienaFit$sf2[ , , -escreen])  100 2
dim(sienaFit$sf2[ , , -escreen])[3]  NA
```

- Why? By default, extraction removes dimensions of length 1.
- Solution? drop=FALSE

```
    dim(sienaFit$sf2[ , , -escreen, drop=FALSE])  100 2 1
dim(sienaFit$sf2[ , , -escreen, drop=FALSE])[3]  1
```

- By default, bipartite networks only have one effect included, hence this error tends to arise for bipartite networks.

Using recover

- Type 1 to inspect the frame of the function with the problem. (If more than one level, usually start at the bottom.)
- Examine the variables that are involved by typing their names:

```
                                  escreen  -99999
                        dim(sienaFit$sf2)  100 2 1
    dim(sienaFit$sf2[ , , -escreen])  100 2
dim(sienaFit$sf2[ , , -escreen])[3]  NA
```

- Why? By default, extraction removes dimensions of length 1.
- Solution? drop=FALSE

```
    dim(sienaFit$sf2[ , , -escreen, drop=FALSE])  100 2 1
dim(sienaFit$sf2[ , , -escreen, drop=FALSE])[3]  1
```

- By default, bipartite networks only have one effect included, hence this error tends to arise for bipartite networks.

Using recover

- Type 1 to inspect the frame of the function with the problem. (If more than one level, usually start at the bottom.)
- Examine the variables that are involved by typing their names:

```
                                  escreen  -99999
                        dim(sienaFit$sf2)  100 2 1
    dim(sienaFit$sf2[ , , -escreen])  100 2
dim(sienaFit$sf2[ , , -escreen])[3]  NA
```

- Why? By default, extraction removes dimensions of length 1.
- Solution? drop=FALSE

```
    dim(sienaFit$sf2[ , , -escreen, drop=FALSE])  100 2 1
dim(sienaFit$sf2[ , , -escreen, drop=FALSE])[3]  1
```

- By default, bipartite networks only have one effect included, hence this error tends to arise for bipartite networks.

Using recover

- Type 1 to inspect the frame of the function with the problem. (If more than one level, usually start at the bottom.)
- Examine the variables that are involved by typing their names:

```
                                  escreen  -99999
                        dim(sienaFit$sf2)  100 2 1
    dim(sienaFit$sf2[ , , -escreen])  100 2
dim(sienaFit$sf2[ , , -escreen])[3]  NA
```

- Why? By default, extraction removes dimensions of length 1.
- Solution? drop=FALSE

```
    dim(sienaFit$sf2[ , , -escreen, drop=FALSE])  100 2 1
dim(sienaFit$sf2[ , , -escreen, drop=FALSE])[3]  1
```

- By default, bipartite networks only have one effect included, hence this error tends to arise for bipartite networks.

Using recover

- Type 1 to inspect the frame of the function with the problem. (If more than one level, usually start at the bottom.)
- Examine the variables that are involved by typing their names:

```
                                  escreen  -99999
                        dim(sienaFit$sf2)  100 2 1
    dim(sienaFit$sf2[ , , -escreen])  100 2
dim(sienaFit$sf2[ , , -escreen])[3]  NA
```

- Why? By default, extraction removes dimensions of length 1.
- Solution? drop=FALSE

```
    dim(sienaFit$sf2[ , , -escreen, drop=FALSE])  100 2 1
dim(sienaFit$sf2[ , , -escreen, drop=FALSE])[3]  1
```

- By default, bipartite networks only have one effect included, hence this error tends to arise for bipartite networks.

Error message from C

Christian found this one.

> *I am trying to run a multigroup analysis which involves main effects of several (time-constant) dyadic covariates. The siena07 function crashes as soon as I include the main effect of a second (not of the first!) dyadic covariate. The error report I get is this:*
>
> *Error in initializeFRAN(z, x, data, effects, prevAns, initC, profileData = profileData, : Dyadic covariate variable 'ADMpath' expected.*
>
> *In this, "ADMpath" is the name of the second dyadic covariate. When running the group-specific projects separately, the error does NOT occur.*

Error message from C: solving it (1)

First find the error message in the code: **grep** or your favourite search method. I use **TextWrangler** on a Mac. In the top RSiena directory, first look in the R:

```
grep "Dyadic covariate variable" R/*
```

Oh dear, not R, now look in the top C/C++ directory:

```
grep "Dyadic covariate variable" src/*
Binary file src/RSiena.dll matches
```

Not useful to me, so try moving down the source directories:

```
grep "Dyadic covariate variable" src/*/*
```

and again

```
grep "Dyadic covariate variable" src/*/*/*
src/model/effects/DyadicCovariateDependentNetworkEffect.cpp:
"Dyadic covariate variable '" + name + "' expected.");
```

Now we have it.

Error message from C: solving it (1)

First find the error message in the code: **grep** or your favourite search method. I use **TextWrangler** on a Mac. In the top RSiena directory, first look in the R:

```
grep "Dyadic covariate variable" R/*
```

Oh dear, not R, now look in the top C/C++ directory:

```
grep "Dyadic covariate variable" src/*
Binary file src/RSiena.dll matches
```

Not useful to me, so try moving down the source directories:

```
grep "Dyadic covariate variable" src/*/*
```

and again

```
grep "Dyadic covariate variable" src/*/*/*
src/model/effects/DyadicCovariateDependentNetworkEffect.cpp:
"Dyadic covariate variable '" + name + "' expected.");
```

Now we have it.

Error message from C: solving it (1)

First find the error message in the code: **grep** or your favourite search method. I use **TextWrangler** on a Mac. In the top RSiena directory, first look in the R:

```
grep "Dyadic covariate variable" R/*
```

Oh dear, not R, now look in the top C/C++ directory:

```
grep "Dyadic covariate variable" src/*
Binary file src/RSiena.dll matches
```

Not useful to me, so try moving down the source directories:

```
grep "Dyadic covariate variable" src/*/*
```

and again

```
grep "Dyadic covariate variable" src/*/*/*
src/model/effects/DyadicCovariateDependentNetworkEffect.cpp:
"Dyadic covariate variable '" + name + "' expected.");
```

Now we have it.

Error message from C: solving it (1)

First find the error message in the code: **grep** or your favourite search method. I use **TextWrangler** on a Mac. In the top RSiena directory, first look in the R:

```
grep "Dyadic covariate variable" R/*
```

Oh dear, not R, now look in the top C/C++ directory:

```
grep "Dyadic covariate variable" src/*
Binary file src/RSiena.dll matches
```

Not useful to me, so try moving down the source directories:

```
grep "Dyadic covariate variable" src/*/*
```

and again

```
grep "Dyadic covariate variable" src/*/*/*
src/model/effects/DyadicCovariateDependentNetworkEffect.cpp:
"Dyadic covariate variable '" + name + "' expected.");
```

Now we have it.

Error message from C: solving it (1)

First find the error message in the code: **grep** or your favourite search method. I use **TextWrangler** on a Mac. In the top RSiena directory, first look in the R:

```
grep "Dyadic covariate variable" R/*
```

Oh dear, not R, now look in the top C/C++ directory:

```
grep "Dyadic covariate variable" src/*
Binary file src/RSiena.dll matches
```

Not useful to me, so try moving down the source directories:

```
grep "Dyadic covariate variable" src/*/*
```

and again

```
grep "Dyadic covariate variable" src/*/*/*
src/model/effects/DyadicCovariateDependentNetworkEffect.cpp:
"Dyadic covariate variable '" + name + "' expected.");
```

Now we have it.

Error message from C: solving it (1)

First find the error message in the code: **grep** or your favourite search method. I use **TextWrangler** on a Mac. In the top RSiena directory, first look in the R:

```
grep "Dyadic covariate variable" R/*
```

Oh dear, not R, now look in the top C/C++ directory:

```
grep "Dyadic covariate variable" src/*
Binary file src/RSiena.dll matches
```

Not useful to me, so try moving down the source directories:

```
grep "Dyadic covariate variable" src/*/*
```

and again

```
grep "Dyadic covariate variable" src/*/*/*
src/model/effects/DyadicCovariateDependentNetworkEffect.cpp:
"Dyadic covariate variable '" + name + "' expected.");
```

Now we have it.

Error message from C: solving it (1)

First find the error message in the code: **grep** or your favourite search method. I use **TextWrangler** on a Mac. In the top RSiena directory, first look in the R:

```
grep "Dyadic covariate variable" R/*
```

Oh dear, not R, now look in the top C/C++ directory:

```
grep "Dyadic covariate variable" src/*
Binary file src/RSiena.dll matches
```

Not useful to me, so try moving down the source directories:

```
grep "Dyadic covariate variable" src/*/*
```

and again

```
grep "Dyadic covariate variable" src/*/*/*
src/model/effects/DyadicCovariateDependentNetworkEffect.cpp:
"Dyadic covariate variable '" + name + "' expected.");
```

Now we have it.

Error message from C: solving it (2)

Now have a look in the code:

```
string name = this->pEffectInfo()->interactionName1();

this->lpConstantCovariate =
    pData->pConstantDyadicCovariate(name);
this->lpChangingCovariate =
    pData->pChangingDyadicCovariate(name);

if(!this->lpConstantCovariate &&!this->lpChangingCovariate)
{
    throw logic_error(
"Dyadic covariate variable '" + name + "' expected.");
}
```

Obviously the matching covariate does not exist. Why not? Let's see what names the covariates are given when they are created.

Error message from C: solving it (2)

Now have a look in the code:

```
string name = this->pEffectInfo()->interactionName1();

this->lpConstantCovariate =
    pData->pConstantDyadicCovariate(name);
this->lpChangingCovariate =
    pData->pChangingDyadicCovariate(name);

if(!this->lpConstantCovariate &&!this->lpChangingCovariate)
{
    throw logic_error(
"Dyadic covariate variable '" + name + "' expected.");
}
```

Obviously the matching covariate does not exist. Why not? Let's see what names the covariates are given when they are created.

Error message from C: solving it (3)

Find the creation code. It will be somewhere in the interface routines which live in the **src** directory. Look for **name** in there.

```
grep names src/siena07*.cpp | less
```

gives a lot of hits, but the only ones near dyadic covariates are in **siena07internals.cpp**. So look in there. In fact in multigroups there are only changing covariates. I find the relevant code, which gets the attribute **name** from the (R) covariate and uses it.

```
SEXP nm;
PROTECT(nm = install("name"));
SEXP name = getAttrib(VECTOR_ELT(VARDYADGROUP,
    changingDyadic), nm);
ChangingDyadicCovariate * pChangingDyadicCovariate =
pData->
createChangingDyadicCovariate(CHAR(STRING_ELT(name, 0)),
myActorSet1, myActorSet2);
```

Error message from C: solving it (3)

Find the creation code. It will be somewhere in the interface routines which live in the **src** directory. Look for **name** in there.

```
grep names src/siena07*.cpp | less
```

gives a lot of hits, but the only ones near dyadic covariates are in **siena07internals.cpp**. So look in there. In fact in multigroups there are only changing covariates. I find the relevant code, which gets the attribute **name** from the (R) covariate and uses it.

```
SEXP nm;
PROTECT(nm = install("name"));
SEXP name = getAttrib(VECTOR_ELT(VARDYADGROUP,
    changingDyadic), nm);
ChangingDyadicCovariate * pChangingDyadicCovariate =
pData->
createChangingDyadicCovariate(CHAR(STRING_ELT(name, 0)),
myActorSet1, myActorSet2);
```

Error message from C: solving it (3)

Find the creation code. It will be somewhere in the interface routines which live in the **src** directory. Look for **name** in there.

```
grep names src/siena07*.cpp | less
```

gives a lot of hits, but the only ones near dyadic covariates are in **siena07internals.cpp**. So look in there. In fact in multigroups there are only changing covariates. I find the relevant code, which gets the attribute **name** from the (R) covariate and uses it.

```
SEXP nm;
PROTECT(nm = install("name"));
SEXP name = getAttrib(VECTOR_ELT(VARDYADGROUP,
    changingDyadic), nm);
ChangingDyadicCovariate *  pChangingDyadicCovariate =
pData->
createChangingDyadicCovariate(CHAR(STRING_ELT(name, 0)),
myActorSet1, myActorSet2);
```

Error message from C: solving it (4)

Now I want to see what this attribute has in it. We could just look in the R object, but as an example, I will examine it in the C. It is an R object in C, so we examine it using **PrintValue**. As a further example we will also look at the created C++ version. For this we use **Rprintf**.

```
SEXP nm;
PROTECT(nm = install("name"));
SEXP name = getAttrib(VECTOR_ELT(VARDYADGROUP,
    changingDyadic), nm);
ChangingDyadicCovariate *  pChangingDyadicCovariate =
pData->
createChangingDyadicCovariate(CHAR(STRING_ELT(name, 0)),
myActorSet1, myActorSet2);
PrintValue(name);
Rprintf("%s\n", pChangingDyadicCovariate->name().c_str());
```

If not already there you need

```
#include <Rinternals.h> for PrintValue
#include <R_ext/Print.h> for Rprintf
```

Error message from C: solving it (4)

Now I want to see what this attribute has in it. We could just look in the R object, but as an example, I will examine it in the C. It is an R object in C, so we examine it using **PrintValue**. As a further example we will also look at the created C++ version. For this we use **Rprintf**.

```
SEXP nm;
PROTECT(nm = install("name"));
SEXP name = getAttrib(VECTOR_ELT(VARDYADGROUP,
    changingDyadic), nm);
ChangingDyadicCovariate *  pChangingDyadicCovariate =
pData->
createChangingDyadicCovariate(CHAR(STRING_ELT(name, 0)),
myActorSet1, myActorSet2);
PrintValue(name);
Rprintf("%s\n", pChangingDyadicCovariate->name().c_str());
```

If not already there you need

```
#include <Rinternals.h> for PrintValue
#include <R_ext/Print.h> for Rprintf
```

Error message from C: solving it (4)

Now we need an example to test it

```
library(RSiena)
mynet <- sienaNet(array(c(s501,s502),dim=c(50,50,2)))
mydy1 <- coDyadCovar(s503)
mydy2 <- coDyadCovar(t(s503))
mydata <- sienaDataCreate(mynet, mydy1, mydy2)
mygrp <- sienaGroupCreate(list(mydata, mydata))
mymodel <- sienaModelCreate()
myeff <- getEffects(mygrp)
myeff <- includeEffects(myeff,X,interaction1='mydy2')
ans <- siena07(mymodel, data=mygrp, effects=myeff)
```

Error message from C: solving it (5)

With the uncorrected code this printed:

```
[1] "mydy1" "mydy2"
mydy1
[1] "mydy1" "mydy2"
mydy1
[1] "mydy1" "mydy2"
mydy1
[1] "mydy1" "mydy2"
mydy1
```

The lines beginning with [1] have come from **`PrintValue`**, the others from **`Rprintf`**. Thus the name attribute is a vector length 2, and the name for every covariate is being set to **`mydy1`**.

Error message from C: solving it (6)

You can check the problem:

```
> attr(mygrp[[1]]$dyvCovars[[1]], 'name')
[1] "mydy1" "mydy2"
> attr(mygrp[[1]]$dyvCovars[[2]], 'name')
[1] "mydy1" "mydy2"
```

Now the code where attribute is set, in `sienaGroupCreate`:

```
const <- objlist[[i]]$dycCovars
for (j in seq(along=const))
{
  attr(newcovar, "mean") <- attr(const[[j]], "mean")
  attr(newcovar, "range") <- attr(const[[j]], "range")
  attr(newcovar, 'name') <- attr(const, "name")
  vars[[nVCovar]] <- newcovar
}
```

It is now fairly obvious there is a `[[j]]` missing.

Error message from C: solving it (6)

You can check the problem:

```
> attr(mygrp[[1]]$dyvCovars[[1]], 'name')
[1] "mydy1" "mydy2"
> attr(mygrp[[1]]$dyvCovars[[2]], 'name')
[1] "mydy1" "mydy2"
```

Now the code where attribute is set, in **sienaGroupCreate**:

```
const <- objlist[[i]]$dycCovars
for (j in seq(along=const))
{
  attr(newcovar, "mean") <- attr(const[[j]], "mean")
  attr(newcovar, "range") <- attr(const[[j]], "range")
  attr(newcovar, 'name') <- attr(const, "name")
  vars[[nVCovar]] <- newcovar
}
```

It is now fairly obvious there is a `[[j]]` missing.

Error message from C: solving it (6)

You can check the problem:

```
> attr(mygrp[[1]]$dyvCovars[[1]], 'name')
[1] "mydy1" "mydy2"
> attr(mygrp[[1]]$dyvCovars[[2]], 'name')
[1] "mydy1" "mydy2"
```

Now the code where attribute is set, in **sienaGroupCreate**:

```
const <- objlist[[i]]$dycCovars
for (j in seq(along=const))
{
  attr(newcovar, "mean") <- attr(const[[j]], "mean")
  attr(newcovar, "range") <- attr(const[[j]], "range")
  attr(newcovar, 'name') <- attr(const, "name")
  vars[[nVCovar]] <- newcovar
}
```

It is now fairly obvious there is a **[[j]]** missing.

New effects