TRANSCRIPT: Data analysis using the RECAP Preterm Platform, ECR module

Andrei Morgan – andrei.morgan@inserm.fr

Explanatory note:
This document contains the transcript for the video, plus some useful reference links that were shown in the video. Additionally, the main demonstration script is included so that you can follow along and/or re-use code in your own analyses. All code is released under the GPL v.3 copyright license – available at: https://www.gnu.org/licenses/gpl-3.0.en.html – and will be published online when the project is finalised.

Introductory slides

Hello, my name is Andrei Morgan, and in this presentation I will show you about performing a data analysis using the RECAP Preterm Platform.

Specifically in this presentation, you will learn about implementing a DataSHIELD analysis on the RECAP Preterm Platform, and I will talk to you a little bit about the differences from the use of normal R where you're analysing data on your own computer to the use of R for federated analyses. Included in this will be a little bit of discussion about the statistical considerations for working with federated data, and I shall also mention a variable naming scheme that we have developed for harmonised data.

As you have all heard already in this module, the demonstration project is looking at the question, "Are there survival differences according to sex in European Extremely preterm populations", and there are five sections to the code. The first is setting up the workspace as you would do normally, you then have to log into the nodes to be able to access the data. The third step is performing the data checks and preparing the data ready for the analysis. We then carry out a descriptive analysis, before moving on to some regression analyses.

For this project, there are a number of different variables that we will be using. For the exposure, we shall use a variable for sex, which we have denoted "sex underscore bin" as it is a binary variable, The outcome is "alive bin discharge" and this is again a binary variable representing whether the subject is alive or not at discharge from hospital. And for the main analyses, we will just use one covariate: gestational age in weeks at birth. I shall also show you some other possibilities that you can do with DataSHIELD, and for this I will use other variables -- specifically, I will use gestational age in days at birth, I shall also use multiple status of the pregnancy, so... the first of all will be multiple status in an integer representing the number of foetuses that were involved in that pregnancy, and I shall then recode it into a binary variable, saying whether it was a singleton or a multiple pregnancy. Finally, I will also use maternal age in years at the birth.

As we've already said, this module provides a demonstration of the abilities of the platform, and the analysis is still in progress, and the results are therefore not final.

Real-time run through

Here I'm going to show you a run through in real time of the code for the demonstration project. On the left hand side of the screen is the script. And on the right hand side of the screen is the output that comes from the console. First of all, we set up the workspace by installing the packages and loading the relevant libraries.

The next stage is to log into the nodes. Here, I am showing you the script, "load all births" that we use to log into the nodes. First of all, we use the function newDSLoginBuilder to create a new object. We add the data for each of the nodes: the data that is required, are the URL; the user; a password -- which here I have loaded from a separate file, so that is not visible; the specific table that we're going to be using; and the driver, which is the Opal driver, And it's the same for all of the nodes. Include this in the object and then, using this object, we build the function, and then we use the function datashield.login to log into the nodes. This takes a few seconds.

The three nodes that we are using represent data sets from ACTION, which is an Italian cohort, and EPICE which is a pan-European cohort, as well as a subsection of the EPICE cohort from Portugal called EPICE-PT. Once the data have been accessed through the nodes, the variables that have been assigned are listed on the screen, as you can see in the output on the right hand side. Now we switch back to the main screen.

Having connected to the node, I want to make sure that it's the "all births" populations that we're using in the working example here. The next thing is to perform some data checks and then to prepare the data. The first thing that we can do is to check the dimensions of the data. Again, this logs into the nodees using the data connections that we've already set up, and it checks the dimensions of each data set. So we can see that, for example, in ACTION, there are 2678 rows and 22 variables. There are more or fewer subject rows in each of the other data sets, but there still are 22 variables and, overall, we have the combined number of rows being 13,981. We can also check the column names. Again, this contacts the nodes, and then it prints out a list of the column names, also known as the variables, in each of the data sets. I have included the list of variables in the script. We then should check some of the information that we have available. Specifically, I know that some of the data sets contain infants or births that occurred at a higher gestational age than the ones that we are interested in. So we should first of all check the gestational age ranges that are included. We do that by tabulating the variable "GA underscore wks underscore birth". And we can see the output on the right hand side. If we switch to that screen temporarily, we can scroll up... And we should see that first of all, the command returns the fact that all the studies are valid and that there are no errors. It then returns the row proportions in the top part of the screen here in the middle. And then the column proportions for each of the data sets. Finally, at the bottom, we have a list of the subject counts per study per week of gestational age. And we can see the gestational age range goes from 22 up to 31 in each of the three studies. The last section of the output shows the combined totals so we first of all get the combined proportions, and then we get the combined source counts for each of the weeks of gestation, with a final message saying the data in all the studies were valid.

We can check the class if we want. We do this to ensure that we're working with a integer. And it is indeed an integer. And this means that we are able to subset using the operator "less than", and this next command ds.DataFrameSubset shows that we take the main data sets - it's a data set that's called capital "D" and we use the variable name "GA underscore WKS underscore birth" with a Boolean operator of "less than" and the second part of the equation is 27 - so gestational age in weeks at birth of less than 27 weeks, and we create a new data set (or a new "object") that is called "baseline".

This has worked in all of the nodes and we get a validity check at the bottom of the output on the right hand side that says that "baseline" appears valid in all sources. However, to double check, we could again look at the size of the data sets. So we can do that by running dimensions ds.dim. In the original data set, if we remind ourselves -- again, it takes a little bit of time -- we see it saw that there were 13,981 rows. And if we check the new object, we can see that this has been reduced down to 4829. If we wanted to check the variable names again of this new object, we see that they're exactly the same as previously.

It's also perhaps interesting to look at the exposure so the exposure in this case is sex, which is meant to be binary.

The first thing to find out is what kind of variable it is. And it is indeed a factor which is what we want.

We then can tabulate the factor. And on the right hand side of the screen we get the outputs. However, here we see that there are in fact three levels to a variable: one, two and nine. Two of the data sets -- the EPICE data sets -- have the value nine whereas the ACTION study has a number of subjects who have missing data. So here we need to correct this because we need the variable to be the same in all of the data sets. The way to do this is slightly complicated in DataSHIELD: we first have to create a numeric vector that includes the same information, and I'm using a command here called ds.asNumeric to create a new object that that is "sex underscore bin dot n" for numeric. Once we have created this variable, we should join it back into the main data frame. That is the next command that I run here using ds.DataFrame, to combine the baseline, and the new vector that we have just created and create a new object or to recreate the object "baseline". To double check that this works, we can again check the dimensions. And if we do that, we now see that there are 23 variables listed for each of the data sets. Because we don't want to include the nines, we can subset the data to exclude the rows that have nine in them. We do this again using the Boolean operator of "less than"... We do that. But the issue we need to be aware of is that the factor levels are still one to a nine, so we need to recode the values of the factor so that we get rid of the nines. And we do that by using ds.recodeValues.

Here, what I've done was to create a new object called "sex underscore bin dot r" to represent recoded. And so if we check the class of that, we will see that it's a factor -- here you go. And if we tabulate it, we can see that now, there are only ones and twos. And we've removed all the missing data. This isn't really a problem for the regression analyses because the regression analyses would remove the NAs, but it is something that we need to bear in mind when we're doing our descriptive analysis... We should merge this variable back into the main data frame.

That works. So to double check it, we could check the name of the new variable in the main data frame. Here, I'm just tabulating the baseline "sex underscore bin dot r", which is the recoded variable. That will give us the same output that we saw a minute ago. And the next command has a cross tabulation of the original "sex underscore bin" variable with the new modified variable that we have just created, this will give us a lot more output. We'll just run that quickly... And if we scroll up on the right hand side of the screen, we start where the command is, we see first of all the information about logging into the nodes, we can see that there are no errors from any of them. And then we see the cross tabulations. First by studies that we see the ACTION row proportions, the column proportions, the EPICE row proportions, the column proportions in EPICE, and then the same thing for EPICE-PT. And then we look at all the combined tables, and we can see all the ones, the proportion of ones that are ones is 100%. And the proportion of twos that are twos is 100%, which is exactly what we want to see, expect to see... And then underneath we get the counts in each of those cross tabulations. And finally, again, the validity message saying that data in all studies were valid.

We should also double check the outcome. The outcome that we're going to use for this is alive at discharge, it is a binary variable, it's a factor. And this time, we need to convert it to numeric for use in a quasipoisson regression that we're going to use later on. So here, I convert it to a numeric variable called "alive bin discharged dot n". And then I include it back into the data frame, as we did previously. And each of these steps takes a little bit of time to run. And it is this that means it's a little bit longer than it would be if you were analysing the data on your own computer, because of the time that it needs to go to the nodes to perform the conversions, and then to return the information to the central analysis computer here. Finally, we're going to look at gestational age here. This time, I'm just checking the class and we see that it's an integer. In fact, we want to study it as a factor so that we can use it and find out the information per gestational, per week of gestational age in the analysis, I'm going to convert it to a factor. And again, after I convert it to a factor, I include it back into the main data frame.

And here I can check the class of the new variable, this is the new variable that is the factor and it is indeed a factor. Something else we could work with we could look at is, for example, multiple status. So here we want to, we want to have a variable that says, whether it's a singleton or multiple pregnancy. However, in this data set, we have "multiple underscore int underscore birth". And what this gives us, it's a factor variable that gives us the number of children that were born in each pregnancy. So we can see that there were singleton's, there were twins, there were triplets, and there are also sets of quadruplets in ACTION and EPICE. We recode this to a binary variable. And here we run into an error. DataSHIELD can be complicated at times, because there are some slightly obscure errors, we could list that, as it suggests, it shows us that there's been an error while trying to run the command on each of the nodes. And if we look at the end, it says that the error is that the variable name text argument is too long. So what we're going to do instead is we're going to rename the variable first of all. So the new variable name is going to be called multiple,

Then we can include it back into the main data frame as before... Then we can check it, as previously. This time I'm doing tabulation first, and then I'm going to look at the class. So we tabulate it, we see the same four levels that we had before and it's still a factor, and this time, we're able to recode it.

And it appears valid in all the data sources. So we check the class. The class is indeed a factor. And so we can tabulate it. And we can see that it has now been recoded into the values of zero and one, representing the Singleton's and the multiple births. And again, finally, we then include that back into the main data frame.

Check it again. Data are still there... we can perform a cross tabulation with the original variable "multiple underscore int underscore birth". If we go over to the right hand side, we're just going to look at the bottom output here for all sources at the very bottom. And we can see that all of those values that were previously coded as two, three or four are now coded as one, which is exactly what we want.

Having done all that, we can now move on to the descriptive analysis. So the first thing that we want to do in our description is to look at the exposure which is sex: I know we've already done this, but we're going to do it again just to get the numbers out of it. And there we have the command ds.table, it prints the information to the screen. What we could also do is we could save that information locally. So we could save it into this object, "tablesexall" that as all the information from that object. Instead, it just prints out the message saying that there were no errors. And if we want the information, we can just run that command and outputs the data that are contained within. We could also just output certain parts of it. So here we're going to take from that object, we're going to take the sex counts, we call that "tablesexcounts", and it's just going to print out the counts from the bottom of the output above. And we can see there, those are the numbers of births in each of the studies that are either sex one or sex two. And in this analysis, the sex one is the males and sex two are the females. We can also output the proportions alone. There are the proportions, there's 53.2% males and 46.8% females. We can also look at the outcome. So the outcome here is survival to discharge,

And we see that there are some missing data for the outcome, in all of the studies, and in all of the studies, it's coded as not available. The outcome otherwise, it's coded as zero for died and one for alive at discharge. We can also do a cross tabulation of the exposure versus the outcome. Similar to before, we specify the row variable and the column variable, it connects to the data sources and it outputs the data on the right hand side. And just looking at the end, we can see that there were 550 males who are alive at discharge, and 458 females who are alive at discharge.

We can also look at other things, so we could look at, for example, the mean of the covariate. So we had gestational age as a numeric vector: we can look at the mean of that using ds.mean. Again, connects to the nodes. And it outputs the estimated mean, the number of data points that it was able to use in the analysis and in the calculation, and the number who were missing if there were any that were missing. In fact all of these data were complete for gestational age. And so we can see that the mean in each of the studies is 23.9 to 24.2, essentially. If we wanted, we could get a combined mean across all of the studies by using this extra option at the end of the command called "type equals combined". And so if we run that, again, it logs into the nodes. And it gives us the estimated mean across all of these studies is 24.1. And that's from 4559 subjects. We could also tabulate these variables if we wanted to just tabulate the gestational age: we can see the combined outputs, or the outputs also by individual studies. And if you remember, we converted this previously into a factor so we should double check that. And we'll see exactly the same results, the same output.

It can also be interesting to plot some of the data. So DataSHIELD provides options for that. We can plot data as a histogram: this is a histogram showing the gestational age in weeks at birth across the three different studies. And what it does is it returns in the "split" form, it returns one graph for each study. And here you can see the histograms for ACTION, EPICE and EPICE-PT. And they're a little bit strange, because they're not continuous along the x-axis, but it does give us an impression of what the spread of the data are, which is quite useful. We return to the other script screen, we can also do it as a "combined" histogram.

And here is that combined histogram of the pooled data using data from all three of the studies. Again, it gives us an idea of what the spread is like. We could also, if we look at the other screen, we can see that it provides a whole load of values as well. And so what we could do is we could take that information and import it or include it in a different type of graph for instance using ggplot or another function using base R If you prefer. Another type of graph that we could do is to perform a scatterplot and these are quite interested in here I'm going to plot gestational age in days against maternal age in years at the birth of the infant. So first of all, we're going to do it for each study separately.

And we can see these three graphs that have been produced -- one for each of the studies -- that show the spread of the data. And DataSHIELD is very clever, because this could potentially be disclosive, and you could identify that there was a child, for instance, at 165 days in the EPICE-PT data set that was born to a mother, of perhaps 19 years of age. But what has happened, and what DataSHIELD does is it adds a slight "fuzzy" factor if you like, that means that the data are just a little bit moved around, so that the data become non-disclosive while retaining the major elements of the data spread across the studies. And again, we could do that as a combined plot. So again, it's going to log into each of the three nodes, it's going to return all the data, and it's going to produce a plot for us, which is here. So there is the plots, and it's quite regular, but we saw that regular pattern in several of the studies, and that that represents the days, the entire days that the infants were born on, with perhaps some fuzziness as well, from the data being spread around to ensure that they are non-disclosive.

The next part of the analysis is the regression analysis. So, as I've said, previously, we're going to take sex, and we're going to look at that in comparison to survival to discharge. And we're taking that as the first example. And the first thing that we want to do is just look at the unadjusted regression analysis. Here we're going to use logistic regression to calculate odds ratios, because the outcome is binary. So we specify in the command we specify the formula, which is alive at discharge according to sex, we say it's family "binomial" and we specify again, the data sources -- and this is very similar to base R, to the glm() function, although it's, it's just a little bit different with the formula being specified manually as opposed to actually being part of the command itself. If we go to the right hand screen, we scroll up and we see that the first thing that happens when we run the command, running the command up here in blue, the first thing that happens is that the information is sent to the nodes, and there are multiple iterations. As the data go to the nodes, the analyses are performed, they're brought back to the main centralised computer and put together and then the, there are subsequent iterations of the data where the results are sent back to the nodes and the combined analysis is used until there is convergence; and eventually we get the convergence, the convergence criterion is true. And we get the information that comes out of it. So we get the intercepts, we get information about the baseline, this is the "information matrix overall" to start with, we get some other information -- for example, we have here, the number of valid subjects that are included; we have the number of missing subjects; and we have the number of total subjects. Then we get a table assessing whether there's a risk of disclosure -- there's no risk of disclosure in this analysis. And there are no errors reported from any of these studies. It took us four iterations, and it's using a binomial family with a link function of the logit, and this is logistic regression. And finally we get the coefficients. What's also interesting amongst this output for the coefficients here is that we see the estimates, we see the standard errors, the Z values, the P values and the 95% confidence intervals. And it then shows us also the exponentiated versions of that, so we get the odds ratios that are given to us. And we can see specifically highlighted here in blue, is the odds ratio for sex two compared to sex one so this is the odds ratio for survival for females compared to males, and the odds ratio is point 9517. And next to that we have the confidence intervals. So here we have the confidence interval that runs from point eight to one point one two.

We could perform an adjusted regression analysis as well taking into consideration gestational age. So we're going to run that quickly. And again, we include that in the function just by using a plus sign, which is shown here on the left. And we add the gestational age in weeks at birth into the function just as you would do in normal R. On the right hand side of the screen now, we can see the output. and here we can see that there is the baseline, and the gestational age in weeks at birth. And we can see the odds ratios. So the odds ratio now for sex is point nine eight, with a confidence range from point eight three to one point one six. And instead, we can also see that the gestational age is in fact very important here. And there is an odds ratio of point six six, with a 95% confidence range from point six one to point seven one. What we could also do is we could use the factor version of this that we use previously to assess this as an interaction. So the interaction between gestational age and between sex. And we have that here on this line with the star instead of the plus sign -- again, exactly as you would do in base R; and we can run this...

It does a number of iterations: this time, it's done five iterations, as opposed to the four that it was doing previously. And it shows us the interaction, which here on the right hand side of the screen, first of all, we have sex, then we have gestational age, and then we have the interaction term here, we can see just looking at the P value, the P value is point one (0.13). And so it's a non-significant interaction.

We could also use the factor variable which was what I really meant earlier is the factor variable to see the interaction per week, as opposed to just with gestational age as a continuous value.

And this time it's interesting, because it's told us that the study data for the EPICE-PT study were invalid: "study data or applied model invalid for this source". Perhaps there were not enough subjects in one of the cells for it to really give us any information. So what we could do instead is we could eliminate that study from the analysis, I've shown that in this next command, and here, for the working data sources, I'm just using one and two, which are the ACTION and EPICE data sets. So these are bigger data sets, so if I run that...

It does a number of iterations again, and then it produces the output to show us the interaction terms. And the results for all of the all of the data. So we have sex, which has a P value of 0.5 so there's actually no difference, no, there's no difference. And then we have gestational age, and then we have the interactions between sex and gestational age that we can figure out.

So I was initially planning to use odds ratios. And we've had a lot of discussion within the group about this. But it's also possible to calculate other things. So we could -- as this is not a rare outcome -- we could instead use risk ratios, and one of the ways of doing that is to use a quasipoisson family. And so I've tried doing that with the same command, this GLM command in DataSHIELD, and it produces errors. And the reason is that, first of all, is that I've been using the main, alive at discharge variable, that was a factor variable. And in fact it needs a numeric outcome to perform this analysis, and so we can try that again....

And this time, it looks like it's working. And it starts to give us some information and then it runs into an error. And the problem is that at the moment, quasipoisson, which is the family that we're trying to use here to calculate risk ratios is not available in DataSHIELD in the GLM function. However, it is available in something called the GLM SLMA function -- so it's the GLM study level meta analysis function. And we can run that command. And what that does is it provides the perspective of the.., it performs the analysis in each of the studies separately, and then it brings the data back after the analysis has been performed, and performs a meta analysis of the individual results from each study. So it's it's three separate results, which are then pooled, as opposed to a single result across the three studies. And in fact, we can also do that with the logistic regression. So just to finish that off, actually, and show you the outputs. There's a lot of output for this function... we scroll up quite a long way. First of all, it performs the analyses, it performs the analyses saving server-side objects. And then it provides a lot of information and a lot of output about the command. And then it gives us results for each study individually. So for example, here are some results for study one, which we know is the ACTION study. And there are a lot of outputs, and eventually, we get the coefficients here at the bottom. So we can see that the estimate is 0.2227. And the P value is 0.0446. It then gives us the outputs for study two, etc. If we jump back to the end, we finally get a combined matrix, or the combined outputs, in fact using the "metafor" package, and it gives us the pooled estimates here, across each of the studies. It gives us three different types of pooled estimates it first of all gives the pooled estimate of each regression coefficient and its standard error with pooling via random effects meta analysis under maximum likelihood, and that's the pooled ML in the first two columns, the pooled and the standard error under maximum likelihood. It then contains estimates and standard errors from random effects meta analysis. So that's the REML columns, the third and fourth column. And finally, it gives us a pooled fixed effects meta analysis, at the end. And we could check what those results actually mean by exponentiating values. So, for example, if we exponentiate the first value, you can see that the relative risk of females compared to males is 1.051, and we could use the standard error to calculate the 95% confidence interval if we wanted as well. And this is using a pooled maximum likelihood function. Similarly, we could we could use the random effects maximum likelihood or the fixed effects as well and see what they give us.

We can do a similar thing, as well, with the logistic regression: we can use a study level meta analysis approach using logistic regression too, so I'll just show that quickly.

Here we have a similar thing: that the pooled estimate is here, and if we exponentiate that, we can see that the estimate is 1.096. And we can calculate the confidence interval as well if we wanted to. And we could compare that if we remember what the previous results were, so we can just do the same thing. So 1.096 is the number to remember. We do the GLM across all of the data sets. And here we see that the odds ratio is 0.9517. But it's not significant and actually the 1.096 fits within the confidence range that goes up to 1.12. So they're similar: they're not the same, but they are similar. I've included some text that is taken from the manual page of the GLM SLMA command from DataSHIELD.

And that's essentially the end of the analysis. So the thing that we then need to do, is we need to log out of the servers. So what we're going to do is we're going to log out of the "all births" servers that we've been using. And that leaves the servers nice and clean for the next time that we want to access them.

Final slide: references
So that concludes the demonstration of the DataSHIELD analysis using the RECAP Preterm Platform. If you want to know more about the demonstration project you can find out information about it on the platform and the link is here. There is also a link to the variable naming scheme, and also to the DataSHIELD software where you can find out more information along with links to the study manual pages of the individual commands, and to a forum where you can ask for more help. If you have any questions, please feel free to contact me either through the forum -- which is the best place because other people will also be able to see the answers to the questions -- or you can contact me directly on my email andrei.morgan@inserm.fr . Thank you.

Links

EPT demonstration project on the RECAP Preterm Platform:

- https://platform.recap-preterm.eu/pub/study/ept-recap

Variable naming scheme:

- https://0xacab.org/asm/cohort_variable_names

DataSHIELD – the analysis software:

- https://www.datashield.org

The main script

```
## RECAP Preterm Summer School - Extreme preterm project demonstration code
## ---------------------------------------------------------------------------
##
## QUESTION: Are there survival differences according to sex in European
## extremely preterm populations?
##
## EXPOSURE: (fetal / infant) sex
##
## OUTCOME(s):
## 1. alive at discharge from hospital
##

## ================================================================================
##
## 1. SET UP WORKSPACE
##
## ================================================================================

## INSTALL PACKAGES ---------------------------------------------------------------
packages <- rownames(installed.packages())

if (!isTRUE("here" %in% packages)) install.packages("here")

if (!isTRUE("DSI" %in% packages)) install.packages("DSI")
if (!isTRUE("DSOpal" %in% packages)) install.packages("DSOpal")
if (!isTRUE("dsBaseClient" %in% packages))
  install.packages("dsBaseClient",
                   repos=c(getOption("repos"),
                           "https://cran.obiba.org"),
                   dependencies=TRUE)


## LOAD LIBRARIES  ----------------------------------------------------------------
loaded_packages <- loadedNamespaces()## ; loaded_packages

if (!isTRUE("here" %in% loaded_packages)) library(here);
if (!isTRUE("DSI" %in% loaded_packages)) library("DSI");
if (!isTRUE("DSOpal" %in% loaded_packages)) library("DSOpal");
if (!isTRUE("dsBaseClient" %in% loaded_packages)) library("dsBaseClient");
packageVersion("dsBaseClient")

## ================================================================================
##
## 2. LOG INTO NODES
##
## ================================================================================

## all births ---------------------------------------------------------------------
## login:
source("load_allbirths.R")

## live births --------------------------------------------------------------------
## login:
source("load_livebirths.R")

## WORKING DATA -------------------------------------------------------------------

## create 'working' connections
working <- allbirths
working <- livebirths
## working  <- nicu
## ==============================================================================
##
## 3. DATA CHECKS AND PREPARATION
##
## ==============================================================================

## check data set size
ds.dim(x="D", datasources= working)

## check column names
ds.colnames(x="D", datasources = working)

## This is the list of variables:
##
## id
## mat_id
## ga_wks_birth
## ga_days_birth
## month_birth
## year_birth
## wgt_g_birth
## multiple_int_birth
## mat_age_yrs_birth
## sex_bin
## alive_bin_onslab
## alive_bin_matadmit
## alive_bin_birth
## alive_bin_nicadmit
## alive_bin_discharge
## steroids_any_bin_antenat
## tocol_any_bin_antenat
## lab_onset_3nom
## delivery_mode_bin
## intub_bin_delivery
## surf_any_bin
## conganom_any_bin

## CHECK DATA SETS -------------------------------------------------------------

## check GA (we want to work with population <27 weeks)
ds.table(rvar="D$ga_wks_birth", datasources = working)

## check class
ds.class("D$ga_wks_birth", datasources = working)

## Subset to less than 27 weeks
ds.dataFrameSubset(
df.name = "D",
V1.name = "D$ga_wks_birth",
V2.name = "27",
Boolean.operator = "<",
newobj = "baseline",
datasources = working)

## check data set size - original
ds.dim(x="D", datasources= working)

## check data set size - new
ds.dim(x="baseline", datasources= working)
## CHECK VARIABLES -------------------------------------------------------------

## check column names again
ds.colnames(x="baseline", datasources = working)
```

```
## EXPOSURE
ds.class(x="baseline$sex_bin", datasources = working)
ds.table(rvar="baseline$sex_bin", datasources = working)

## There's a problem later with the factor classes, so we need to get
## rid of '9' which some studies have used to code missing. But to do
## that, we need to convert to numeric first.
ds.asNumeric(x.name="baseline$sex_bin",
             newobj= "sex_bin.n",
             datasources = working)

## join it back into main dataframe
ds.dataFrame(x=c("baseline","sex_bin.n"),
             newobj = "baseline",
             datasources = working)

## check dimensions - we see there is an extra variable now
ds.dim(x="baseline", datasources=working)

## Subset to sex to less than 9
ds.dataFrameSubset(
df.name = "baseline",
V1.name = "baseline$sex_bin.n",
V2.name = "9",
Boolean.operator = "<",
newobj = "baseline",
datasources = working)

## issue was the coding (1,2,9) so need to recode 9 = NA
ds.recodeValues(
  var.name = "baseline$sex_bin",
  values2replace = c(1,2,9),
  new.values.vector = c(1,2,NA),
  newobj = "sex_bin.r",
  datasources = working
)

ds.class(x="sex_bin.r", datasources = working)
ds.table(rvar="sex_bin.r", datasources = working)

## merge back into main data frame
ds.dataFrame(x=c("baseline","sex_bin.r"),
             newobj = "baseline",
             datasources = working)

ds.table(rvar="baseline$sex_bin.r", datasources = working)
ds.table(rvar="baseline$sex_bin", cvar="baseline$sex_bin.r",
         datasources = working)

## OUTCOME(S)
ds.class(x="baseline$alive_bin_discharge", datasources = working)

## This time, we need to convert to numeric for use in 'quasipoisson' family
ds.asNumeric(x.name="baseline$alive_bin_discharge",
            newobj = "alive_bin_discharge.n",
            datasources = working)
ds.dataFrame(x=c("baseline","alive_bin_discharge.n"),
             newobj = "baseline",
             datasources = working)

## GESTATIONAL AGE
ds.class(x="baseline$ga_wks_birth", datasources = working)

## convert to factor
```

```
ds.asFactor(input.var.name="baseline$ga_wks_birth",
            newobj.name = "ga_wks_birth.f",
            datasources = working)
ds.dataFrame(x=c("baseline","ga_wks_birth.f"),
             newobj = "baseline",
             datasources = working)

## double check
ds.class(x="baseline$ga_wks_birth.f", datasources = working)

## MULTIPLE STATUS
ds.class(x="baseline$multiple_int_birth", datasources = working)
ds.table(rvar="baseline$multiple_int_birth", datasources = working)

## recode to binary variable
ds.recodeValues(
  var.name = "baseline$multiple_int_birth",
  values2replace.vector = c(1,2,3,4),
  new.values.vector = c(0,1,1,1),
  newobj = "multiple_bin_birth",
  datasources = working
)

## Didn't work because name is too long, so rename variable first
ds.assign(toAssign = "baseline$multiple_int_birth",
          newobj = "multiple",
          datasources = working)

## include back in dataframe
ds.dataFrame(x=c("baseline","multiple"),
             newobj = "baseline",
             datasources = working)

## check it
ds.table(rvar="multiple", datasources = working)
ds.class(x="multiple", datasources = working)

## now can recode it
ds.recodeValues(
  var.name = "baseline$multiple",
  values2replace.vector = c(1,2,3,4),
  new.values.vector = c(0,1,1,1),
  newobj = "multiple_bin",
  datasources = working
)

ds.class("multiple_bin", datasources = working)
ds.table(rvar="multiple_bin", datasources = working)

## include back in data frame
ds.dataFrame(x=c("baseline","multiple_bin"),
             newobj = "baseline",
             datasources = working)

## and check it
ds.table(rvar="baseline$multiple_bin",
         datasources = working)

## cross tab
ds.table(cvar="baseline$multiple_bin",
         rvar="baseline$multiple_int_birth",
         datasources = working)

## ============================================================================
##
## 4. DESCRIPTIVE ANALYSIS
##
## ==============================================================================

## exposure = SEX
ds.table(rvar="baseline$sex_bin.r", datasources = working)

## save locally
tablesexall <- ds.table(rvar="baseline$sex_bin.r", datasources = working)
tablesexall

## obtain just the counts - local object
tablesexcounts <- tablesexall$output.list$TABLE_rvar.by.study_counts
tablesexcounts

## other local things - e.g. combined proportions
tablesexall$output.list$TABLES.COMBINED_all.sources_proportions


## outcome = SURVIVAL TO DISCHARGE
ds.table(rvar="baseline$alive_bin_discharge", datasources = working)

## cross tab (exposure v outcome)
ds.table(rvar="baseline$sex_bin.r",
         cvar="baseline$alive_bin_discharge", datasources = working)

## co-variate
ds.mean(x="baseline$ga_wks_birth", datasources = working)
ds.mean(x="baseline$ga_wks_birth", datasources = working, type="combined")
ds.table(rvar="baseline$ga_wks_birth", datasources = working)

## co-variate - factor, should be the same!
ds.table(rvar="baseline$ga_wks_birth.f", datasources = working)

ds.histogram(x="baseline$ga_wks_birth", type = "split",
             datasources = working)

ds.histogram(x="baseline$ga_wks_birth", type = "combine",
             datasources = working)

ds.scatterPlot(x="baseline$ga_days_birth",
               y="baseline$mat_age_yrs_birth",
               datasources = working)

ds.scatterPlot(x="baseline$ga_days_birth",
               y="baseline$mat_age_yrs_birth",
               type="combine",
               datasources = working)

## ==============================================================================
##
## 5. REGRESSION ANALYSIS
##
## ==============================================================================

## SEX (all births) v. SURVIVAL TO DISCHARGE - taken as a first analysis as a
## test/example

## ------------------------------------------------------------------------------

## crude association (unadjusted) -- logistic regression
ds.glm(formula="baseline$alive_bin_discharge ~ baseline$sex_bin.r",
       family="binomial", datasource = working)
```

```
## adjusted association (GA) -- logistic regression
ds.glm(formula="baseline$alive_bin_discharge ~ baseline$sex_bin.r +
      baseline$ga_wks_birth",
       family="binomial", datasource = working)

## adjusted association (GA: interaction) -- logistic regression
ds.glm(formula="baseline$alive_bin_discharge ~ baseline$sex_bin.r *
      baseline$ga_wks_birth",
       family="binomial", datasource = working)

## adjusted association (GA: interaction (factors) -- logistic
## regression
ds.glm(formula="baseline$alive_bin_discharge ~ baseline$sex_bin.r *
      baseline$ga_wks_birth.f",
       family="binomial", datasource = working)

## adjusted association (GA: interaction (factors) -- logistic
## regression (w/o EPICE-PT)
ds.glm(formula="baseline$alive_bin_discharge ~ baseline$sex_bin.r *
      baseline$ga_wks_birth.f",
       family="binomial", datasource = working[c(1,2)])

## Initially, I was planning to calculate odds ratios, but given that
## the outcome (death) is not rare, it might make more sense to
## calculate risk ratios. This can be done use 'quasipoisson':
##
## http://freerangestats.info/blog/2018/08/17/risk-ratios
##

## crude association (unadjusted) -- quasipoisson regression
ds.glm(formula="baseline$alive_bin_discharge ~ baseline$sex_bin.r",
       family="quasipoisson", datasource = working)

## that doesn't work as requires numeric outcome:
ds.glm(formula="baseline$alive_bin_discharge.n ~ baseline$sex_bin.r",
       family="quasipoisson", datasource = working)
## And that doesn't work as 'quasipoisson' isn't implemented yet for
## combined analyses....

## We therefore need to use the SLMA version.
ds.glmSLMA(formula=alive_bin_discharge.n ~ sex_bin.r,
           dataName = "baseline",
           family="quasipoisson", datasource = working)

## What about SLMA logistic regression
ds.glmSLMA(formula=alive_bin_discharge.n ~ sex_bin.r,
           dataName = "baseline",
           family="binomial", datasource = working)

## How does it compare to the basic (IPD) glm?
ds.glm(formula="baseline$alive_bin_discharge ~ baseline$sex_bin.r",
           family="binomial", datasource = working)

## NB there is good documentation for ds.glmSLMA:
##
## https://rdrr.io/github/datashield/dsBaseClient/man/ds.glmSLMA.html
##
## Specifically, of note is the following (taken from the second
## paragraph of the 'Details'):
##
## Although the ds.glm approach might at first sight appear to be
## preferable under all circumstances, this is not always the
## case. First, the results from both approaches are generally very
## similar. Secondly, the SLMA approach can offer key inferential
## advantages when there is marked heterogeneity between sources that
## cannot simply be corrected by including fixed-effects in one's ds.glm
## model that each reflect a study- or centre-specific effect. In
## particular, such fixed effects cannot be guaranteed to generate
## formal inferences that are unbiased when there is heterogeneity in
## the effect that is actually of scientific interest. It might be
## argued that one should not try to pool the inferences anyway if there
## is marked heterogeneity, but you can use the joint analysis to
## formally check for such heterogeneity and then choose to report the
## pooled result or separate results from each study
## individually. Crucially, unless the heterogeneity is substantial,
## pooling can be quite reasonable. Furthermore, if you just fit a
## ds.glm model without centre-effects you will in effect be pooling
## across all studies without checking for heterogeneity and if
## heterogeneity exists and if it is strong you can get theoretically
## results that are badly confounded by study.
##
## * * *

## LOGOUT OF SERVERS ---------------------------------------------------------------

## datashield.logout(connections)
datashield.logout(allbirths)
datashield.logout(livebirths)
datashield.logout(nicu)
```