Help! My code is broken

Sarah Watkins

Help! My code is broken

Sarah Watkins

Aims of this session

- Build on the broken code session of part 1

Aims of this session

- Build on the broken code session of part 1
- Firstly, we will look at the most common errors in R

Aims of this session

- Build on the broken code session of part 1
- Firstly, we will look at the most common errors in R
- Secondly, we have an analysis that we will walk through together
  - Some of the code will break!
  - We will look at how we can work out what to fix
  - We hope this will give you practical experience and more confidence in how to fix your code

Common errors in R and how to interpret them

https://github.com/noamross/zero-dependency-problems/blob/master/misc/stack-overflow-common-r-errors.md

R error messages are notoriously hard to interpret

R error messages are notoriously hard to interpret

- …and answers on forums can be equally difficult

R error messages are notoriously hard to interpret

- …and answers on forums can be equally difficult
- Noam Ross has looked at the most common error messages in R and given a brief simple description of what they mean

R error messages are notoriously hard to interpret

- …and answers on forums can be equally difficult
- Noam Ross has looked at the most common error messages in R and given a brief simple description of what they mean
- The descriptions are not specific but will let you know what aspect of your data or code you need to check

“tl; dr: Most errors in R are due to looking for something that isn't there”

“tl; dr: Most errors in R are due to looking for something that isn't there”

- Used R to derive errors posted to Stack Overflow

“tl; dr: Most errors in R are due to looking for something that isn't there”

- Used R to derive errors posted to Stack Overflow
- >10,000 errors

“tl; dr: Most errors in R are due to looking for something that isn't there”

- Used R to derive errors posted to Stack Overflow
- >10,000 errors

```
##    trigram                              freq
## 1 not find function                     311
## 2 Error in if                           308
## 3 could not find                        300
## 4 in eval(expr, envir,                  298
## 5 eval(expr, envir, enclos)             291
## 6 envir, enclos) :                      287
## 7 Error in eval(expr,                   286
## 8 = TRUE) :                             249
## 9 value where TRUE/FALSE                239
## 10 missing value where                  230
## 11 : cannot open                        216
## 12 : missing value                      214
## 13 enclos) : object                     211
## 14 where TRUE/FALSE needed              201
## 15 : unable to                          194
```

"could not find function"

- Usually either the package is not loaded, or the function has been misspelled

"Error in if"

- generally means the logical statement in "if (XXX) { ..." is not yielding a logical value
- Most of these have missing value where TRUE/FALSE needed, **meaning that the variable in XXX has NA in it**.

"Error in eval"

- caused by references to objects that don't exist

"object not found errors"

- the user has written a statement that's looking for an object not in memory
- Have you misspelled the object?
- Have you created the necessary object?
- Have you loaded the object?

“subscript out of bounds”

- These errors occur when you try to access an element of a vector or list, or a dimension, that isn't there
- I most often get this error if I’ve missed a comma (specifying rows or columns) or erroneously included a comma

“replacement has”

- attempt to assign a vector of values to a subset of an existing object when the lengths do not match up
- Check vector length + object dimensions – do they match?

Summary of part 1

- We often encounter the same errors in our code!
- These will differ for everyone (depends what you are doing/what packages you tend to use)
- Practical tip: make a spreadsheet (or similar) of the errors you encounter and what you did to resolve them; this could help you solve errors quicker in the future

Part 2: Broken code walkthrough

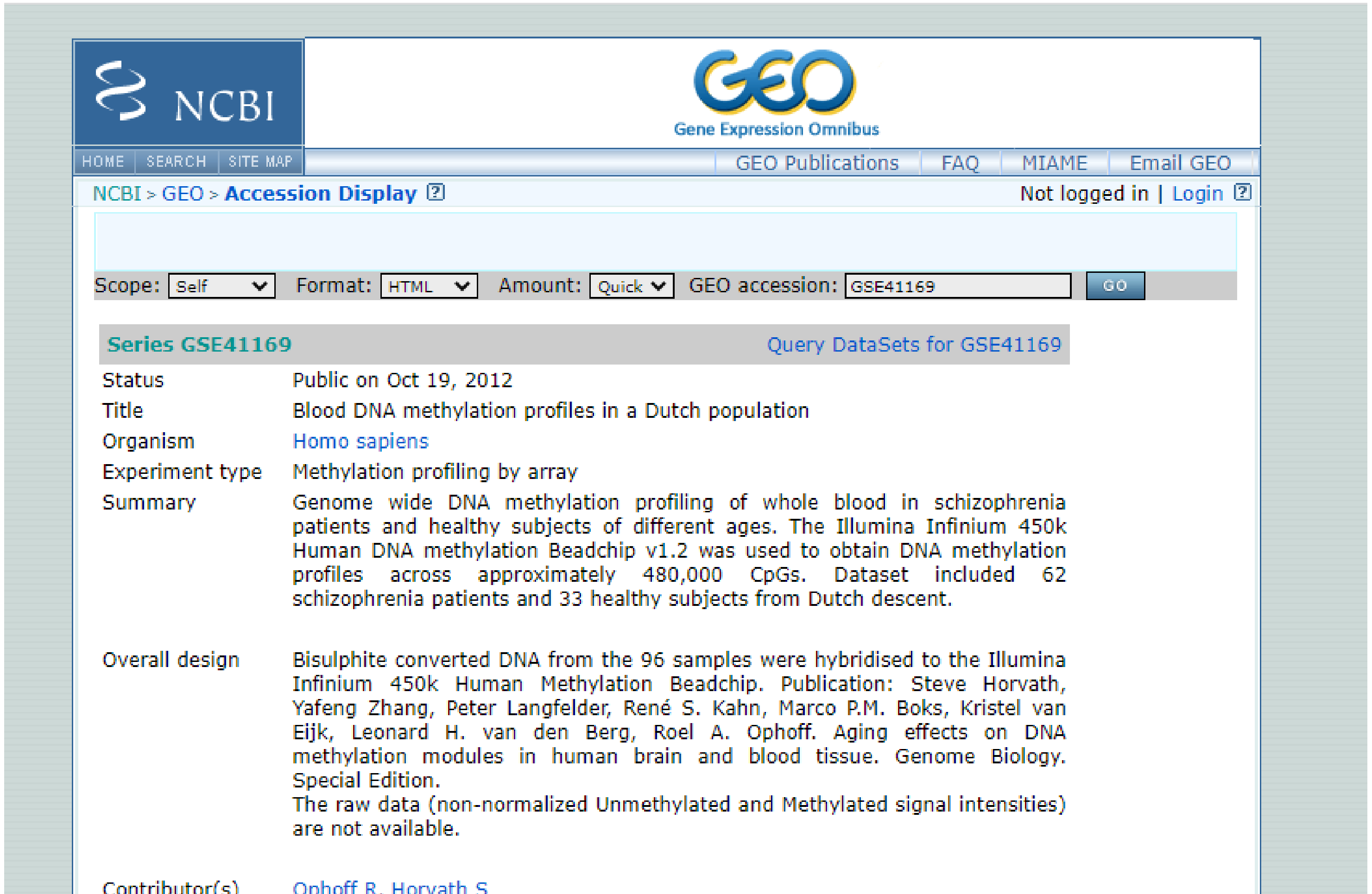
Let’s use some of R’s functions to help us fix some code

Introduction to our analysis

- Today we will use a publicly available dataset (GSE41169)

Introduction to our analysis

- Today we will use a publicly available dataset (GSE41169)



Scope: Self Format: HTML Amount: Quick GEO accession: GSE41169

Series GSE41169 Query DataSets for GSE41169

Status	Public on Oct 19, 2012
Title	Blood DNA methylation profiles in a Dutch population
Organism	Homo sapiens
Experiment type	Methylation profiling by array
Summary	Genome wide DNA methylation profiling of whole blood in schizophrenia patients and healthy subjects of different ages. The Illumina Infinium 450k Human DNA methylation Beadchip v1.2 was used to obtain DNA methylation profiles across approximately 480,000 CpGs. Dataset included 62 schizophrenia patients and 33 healthy subjects from Dutch descent.
Overall design	Bisulphite converted DNA from the 96 samples were hybridised to the Illumina Infinium 450k Human Methylation Beadchip. Publication: Steve Horvath, Yafeng Zhang, Peter Langfelder, René S. Kahn, Marco P.M. Boks, Kristel van Eijk, Leonard H. van den Berg, Roel A. Ophoff. Aging effects on DNA methylation modules in human brain and blood tissue. Genome Biology. Special Edition. The raw data (non-normalized Unmethylated and Methylated signal intensities) are not available.
Contributor(s)	Ophoff R, Horvath S

Introduction to our analysis

- Today we will use a publicly available dataset (GSE41169)
- We have phenotype and DNA methylation data for 95 participants

Introduction to our analysis

- Today we will use a publicly available dataset (GSE41169)
- We have phenotype and DNA methylation data for 95 participants
- I have edited the dataset slightly (so don't use this version for other projects!)

Introduction to our analysis

- Today we will use a publicly available dataset (GSE41169)
- We have phenotype and DNA methylation data for 95 participants
- I have edited the dataset slightly (so don’t use this version for other projects!)
- We have only kept 10 DNAm sites to keep this simple

Introduction to our analysis

- Today we will use a publicly available dataset (GSE41169)
- We have phenotype and DNA methylation data for 95 participants
- I have edited the dataset slightly (so don’t use this version for other projects!)
- We have only kept 10 DNAm sites to keep this simple
- This dataset contains DNAm for controls and individuals with schizophrenia

1. Reading in the data

- The datasets are already in R files so you can load by filling in your own file path:

```
load("path/to/samplesheet/GSE41169_samplesheet_intro_to_R.Rdata“)
load("path/to/meth/GSE41169_matrix_intro_to_R.Rdata")
```

1. Reading in the data

- The datasets are already in R files so you can load by filling in your own file path:

```
load("path/to/samplesheet/GSE41169_samplesheet_intro_to_R.Rdata")
load("path/to/meth/GSE41169_matrix_intro_to_R.Rdata")
```

- Let’s have a look at our data:

```
colnames(samplesheet)
dim(samplesheet)
dim(meth)
```

What’s the mean age of our participants?

```
mean(samplesheet$numericage)
```

What’s the mean age of our participants?

```
mean(samplesheet$numericage)
NA
```

What’s the mean age of our participants?

```
mean(samplesheet$numericage)
```

NA

Let’s use debug() to find out what the problem might be

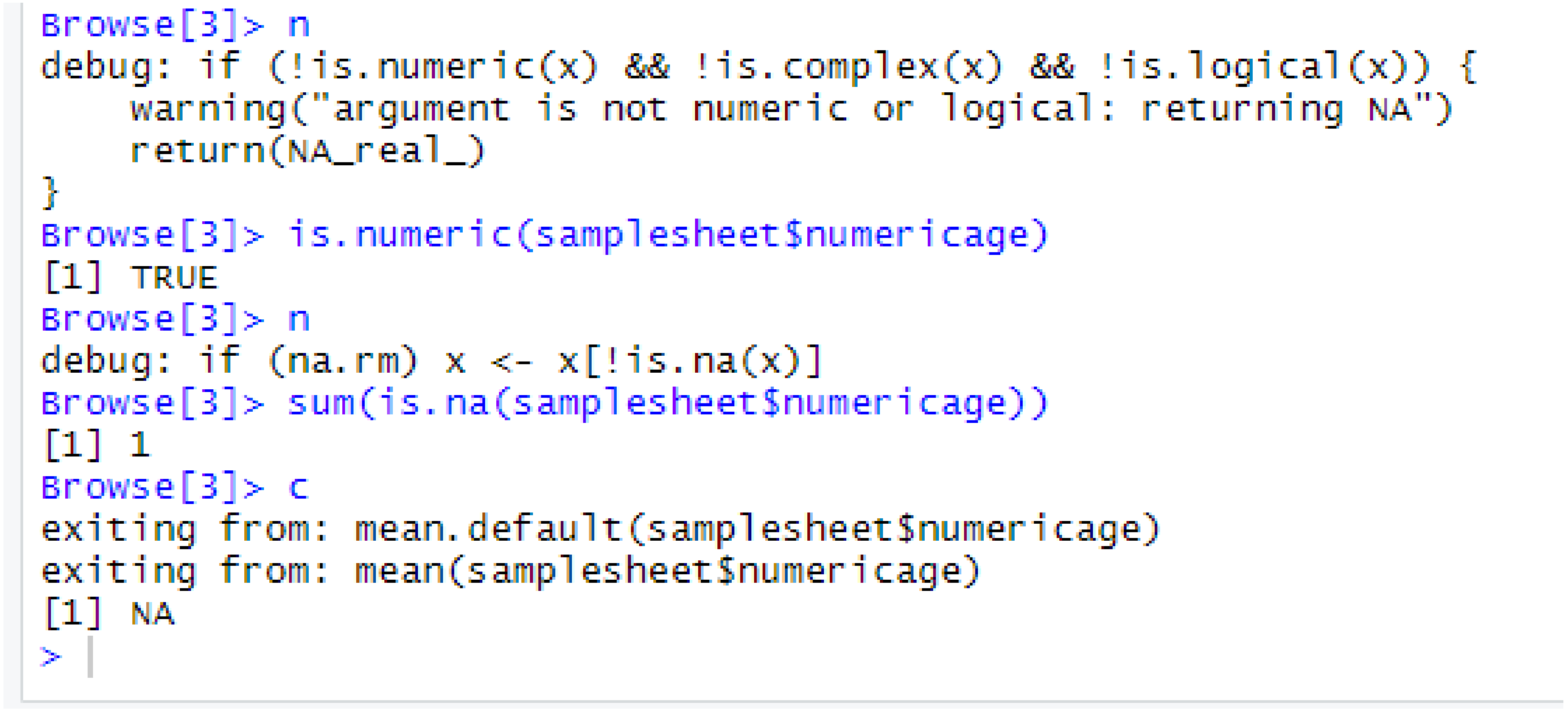
```
debug(mean)
```

```
mean(samplesheet$numericage)
```

We can browse through each line of the mean() function, moving to the next by entering n and then pressing enter

In each section, we can test our data

What’s the mean age of our participants?



```
Browse[3]> n
debug: if (!is.numeric(x) && !is.complex(x) && !is.logical(x)) {
    warning("argument is not numeric or logical: returning NA")
    return(NA_real_)
}
Browse[3]> is.numeric(samplesheet$numericage)
[1] TRUE
Browse[3]> n
debug: if (na.rm) x <- x[!is.na(x)]
Browse[3]> sum(is.na(samplesheet$numericage))
[1] 1
Browse[3]> c
exiting from: mean.default(samplesheet$numericage)
exiting from: mean(samplesheet$numericage)
[1] NA
> 
```

What’s the mean age of our participants?

Make sure you exit the debugger using undebug()

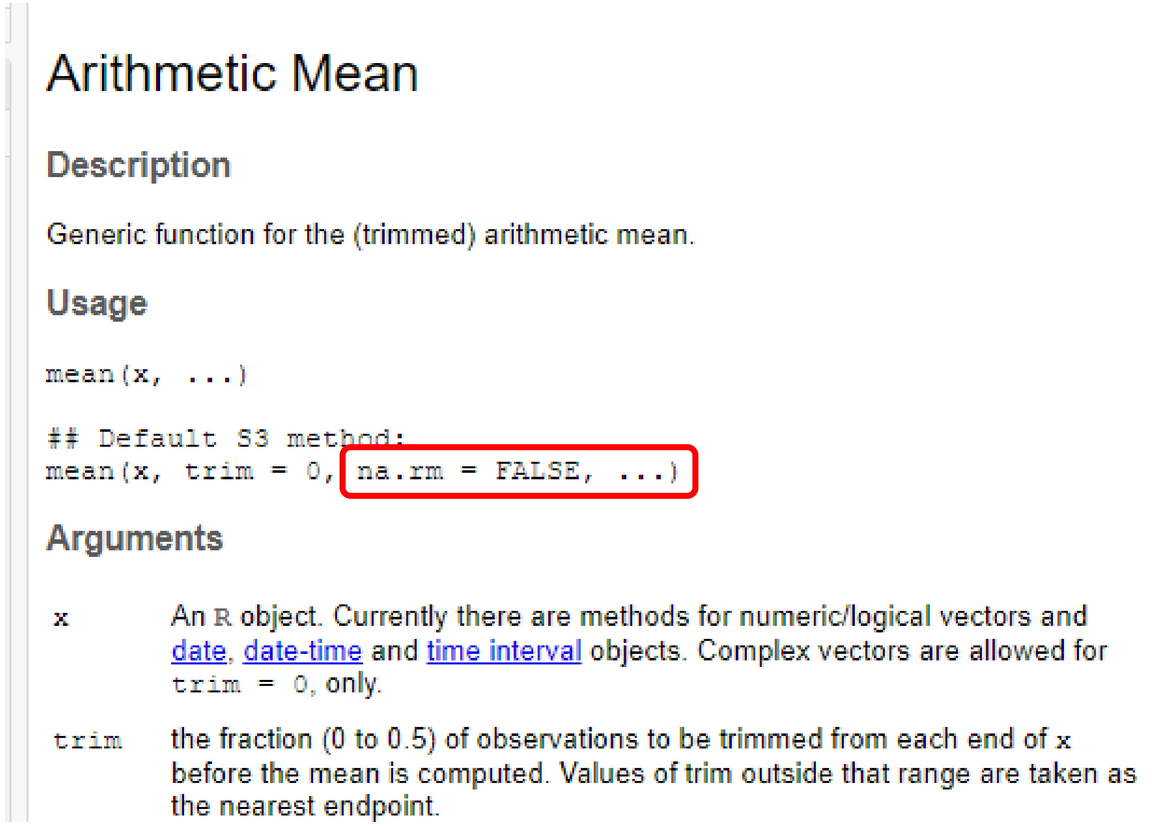
```
undebug(mean)
```

How do we remove Nas using mean()?

```
?mean
```

So to run mean without Nas:

```
mean(samplesheet$numericage,n
```



Arithmetic Mean

Description

Generic function for the (trimmed) arithmetic mean.

Usage

```
mean(x, ...)

## Default S3 method:
mean(x, trim = 0, na.rm = FALSE, ...)
```

Arguments

`x`	An R object. Currently there are methods for numeric/logical vectors and date, date-time and time interval objects. Complex vectors are allowed for `trim = 0`, only.
`trim`	the fraction (0 to 0.5) of observations to be trimmed from each end of `x` before the mean is computed. Values of trim outside that range are taken as the nearest endpoint.

Example 2: Let’s do a regression

- We want to test a regression between disease status and DNA methylation using the first 3 DNAm sites
- Let’s test one DNAm site to begin with

```
temp<-summary(lm(meth[,1]~samplesheet$diseasestatus2))
```

Let’s do a regression

- We want to test a regression between disease status and DNA methylation using the first 3 DNAm sites
- Let’s test one DNAm site to begin with

```
temp<-summary(lm(meth[,1]~samplesheet$diseasestatus2))
```

```
Error in model.frame.default(formula = meth[, 1] ~
samplesheet$diseasestatus2, : variable lengths differ (found
for 'samplesheet$diseasestatus2')
```

Let’s do a regression

- We want to test a regression between disease status and DNA methylation using the first 3 DNAm sites
- Let’s test one DNAm site to begin with

```
temp<-summary(lm(meth[,1]~samplesheet$diseasestatus2))
```

```
Error in model.frame.default(formula = meth[, 1] ~ 
samplesheet$diseasestatus2, : variable lengths differ (found 
for 'samplesheet$diseasestatus2')
```

```
dim(meth)
```

```
meth[1:5,1:5]
```

Let’s try that again

```
meth <- t(meth)
```

Let’s try that again

```
meth <- t(meth)
dim(meth)
```

Let’s try that again

```
meth <- t(meth)
dim(meth)
temp <-
summary(lm(meth[,1]~samplesheet$diseasestatus2))
```

Now let’s test whether DNAm is associated with some of our covariates

```
exposures <- c("numericage","plate","position")
```

Now let’s test whether DNAm is associated with some of our covariates

```
exposures <- c("numericage","plate","position")
```

- Create a list to save our output to

```
results_list <- list()
```

Now let’s test whether DNAm is associated with some of our covariates

```
exposures <- c("numericage","plate","position")
```

- Create a list to save our output to

```
results_list <- list()
```

- And run a loop to test the association with DNAm

```
for(i in exposures){
  temp <- lm(meth~samplesheet[,i])
  results_list[[i]] <- temp
}
```

Now let's test whether DNAm is associated with some of our covariates

```
exposures <- c("numericage","plate","position")
```

- Create a list to save our output to

```
results_list <- list()
```

- And run a loop to test the association with DNAm

```
for(i in exposures){
  temp <- lm(meth~samplesheet[,i])
  results_list[[i]] <- temp
}
```

```
Error in `contrasts<-`(`*tmp*`, value = contr.funs[1 + isOF[nn]]) :
contrasts can be applied only to factors with 2 or more levels
```

Where is that error happening?

- We can use traceback() right after the code that didn’t work:

Where is that error happening?

- We can use traceback() right after the code that didn't work:

```
traceback()
```

```
5: stop("contrasts can be applied only to factors with 2 or more levels")
4: `contrasts<-`(`*tmp*`, value = contr.funs[1 + isOF[nn]])
3: model.matrix.default(mt, mf, contrasts)
2: model.matrix(mt, mf, contrasts)
1: lm(meth ~ samplesheet[, i])
```

Where is that error happening?

- We can use traceback() right after the code that didn't work:

```
traceback()
```

```
5: stop("contrasts can be applied only to factors with 2 or more levels")
4: `contrasts<-`(`*tmp*`, value = contr.funs[1 + isOF[nn]])
3: model.matrix.default(mt, mf, contrasts)
2: model.matrix(mt, mf, contrasts)
1: lm(meth ~ samplesheet[, i])
```

- So we know that there's a problem with the lm() in the loop

Finding out which loop variable is causing the problem

- We can use print() within the loop to find out which variable it’s breaking on:

Finding out which loop variable is causing the problem

- We can use print() within the loop to find out which variable it’s breaking on:

```
for(i in exposures){
  print(i)
  temp <- lm(meth~samplesheet[,i])
  results_list[[i]] <- temp
}
```

Finding out which loop variable is causing the problem

- We can use print() within the loop to find out which variable it’s breaking on:

```
for(i in exposures){
  print(i)
  temp <- lm(meth~samplesheet[,i])
  results_list[[i]] <- temp
}
```

```
[1] "numericage"
[1] "plate"
Error in `contrasts<-`(`*tmp*`, value = contr.funs[1 + isOF[nn]]) :
contrasts can be applied only to factors with 2 or more levels
```

Ok, we know the loop is breaking on the ‘plate’ variable

- What do we think the error suggests?

```
Error in `contrasts<-`(`*tmp*`, value = contr.funs[1 + isOF[nn]]) :
contrasts can be applied only to factors with 2 or more levels
```

Ok, we know the loop is breaking on the ‘plate’ variable

- What do we think the error suggests?

```
Error in `contrasts<-`(`*tmp*`, value = contr.funs[1 + isOF[nn]]) :
contrasts can be applied only to factors with 2 or more levels
```

- Plate might not have 2 or more levels!

Ok, we know the loop is breaking on the ‘plate’ variable

- What do we think the error suggests?

```
Error in `contrasts<-`(`*tmp*`, value = contr.funs[1 + isOF[nn]]) :
contrasts can be applied only to factors with 2 or more levels
```

- Plate might not have 2 or more levels!
- We can test this with table()

Ok, we know the loop is breaking on the ‘plate’ variable

- What do we think the error suggests?

```
Error in `contrasts<-`(`*tmp*`, value = contr.funs[1 + isOF[nn]]) :
contrasts can be applied only to factors with 2 or more levels
```

- Plate might not have 2 or more levels!
- We can test this with table()

```
table(samplesheet$plate)
```

Ok, we know the loop is breaking on the 'plate' variable

- What do we think the error suggests?

```
Error in `contrasts<-`(`*tmp*`, value = contr.funs[1 + isOF[nn]]) :
contrasts can be applied only to factors with 2 or more levels
```

- Plate might not have 2 or more levels!
- We can test this with table()

```
table(samplesheet$plate)

   plate: PLATE A
               95
```

Summary

Summary

- Errors are usually due to R looking for something that isn’t there
- If you get an error, check the data going in to the function
  - Are the objects there, are they spelled correctly, check the length/dimensions, check for NAs, etc

Summary

- Errors are usually due to R looking for something that isn’t there
- If you get an error, check the data going in to the function
  - Are the objects there, are they spelled correctly, check the length/dimensions, check for NAs, etc
- We hope that you have learned a few tricks to investigate the source of coding errors
- We hope you feel a bit more confident in trying to find the source of errors
- Was this helpful? Let us know!