TROUBLESHOOTING R

Isolating Issues and Asking Questions

Erik Kusch

erik.kusch@au.dk

Section for Ecoinformatics & Biodiversity
Center for Biodiversity and Dynamics in a Changing World (BIOCHANGE)
Aarhus University

17/02/2021

So your analysis is misbehaving. . .

Start by **investigating common errror sources**:

Code Errors **Mis-Specified Statistics** **Unexpected Findings**

So your analysis is misbehaving. . .

Start by **investigating common errror sources**:

Code Errors **Mis-Specified Statistics** **Unexpected Findings**

So your analysis is misbehaving...

Start by **investigating common errror sources**:

Code Errors

- Capitalisation
- Mismatched...
  - Object types
  - Value modes
  - Object dimensions
- Directory structures
- Non-self-contained scripts

Mis-Specified Statistics

Unexpected Findings

So your analysis is misbehaving. . .

Start by **investigating common errror sources**:

Code Errors

- Capitalisation
- Mismatched...
  - Object types
  - Value modes
  - Object dimensions
- Directory structures
- Non-self-contained scripts

Mis-Specified Statistics

- Assumptions not met
- Erroneous outliers
- Mis-interpretation of outputs
- Incorrect data handling

Unexpected Findings

So your analysis is misbehaving...

Start by **investigating common errror sources**:

Code Errors

- Capitalisation
- Mismatched...
  - Object types
  - Value modes
  - Object dimensions
- Directory structures
- Non-self-contained scripts

Mis-Specified Statistics

- Assumptions not met
- Erroneous outliers
- Mis-interpretation of outputs
- Incorrect data handling

Unexpected Findings

"[...] in an experimental framework, surprising results are significant results, rather than signs of failure."

$\sim$ Curtin & Parker, 2014

So your analysis is misbehaving. . .

Start by **investigating common errror sources**:

Code Errors

- Capitalisation
- Mismatched...
  - Object types
  - Value modes
  - Object dimensions
- Directory structures
- Non-self-contained scripts

Mis-Specified Statistics

- Assumptions not met
- Erroneous outliers
- Mis-interpretation of outputs
- Incorrect data handling

Unexpected Findings

"[...] in an experimental framework, surprising results are significant results, rather than signs of failure."

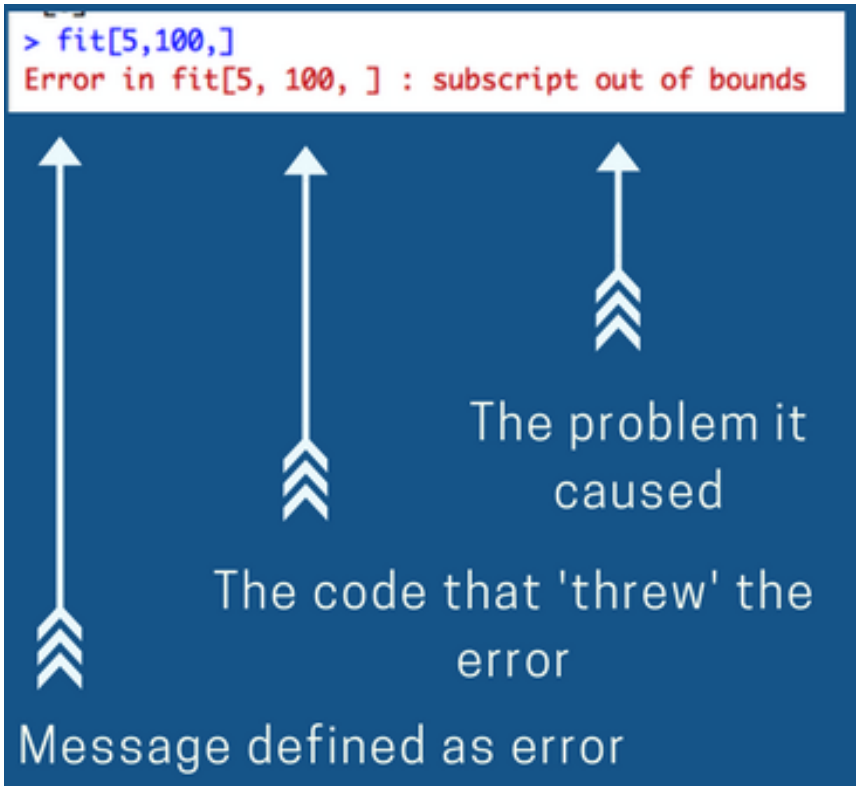
$\sim$ Curtin & Parker, 2014

How do we resolve these?

Errors and Warnings

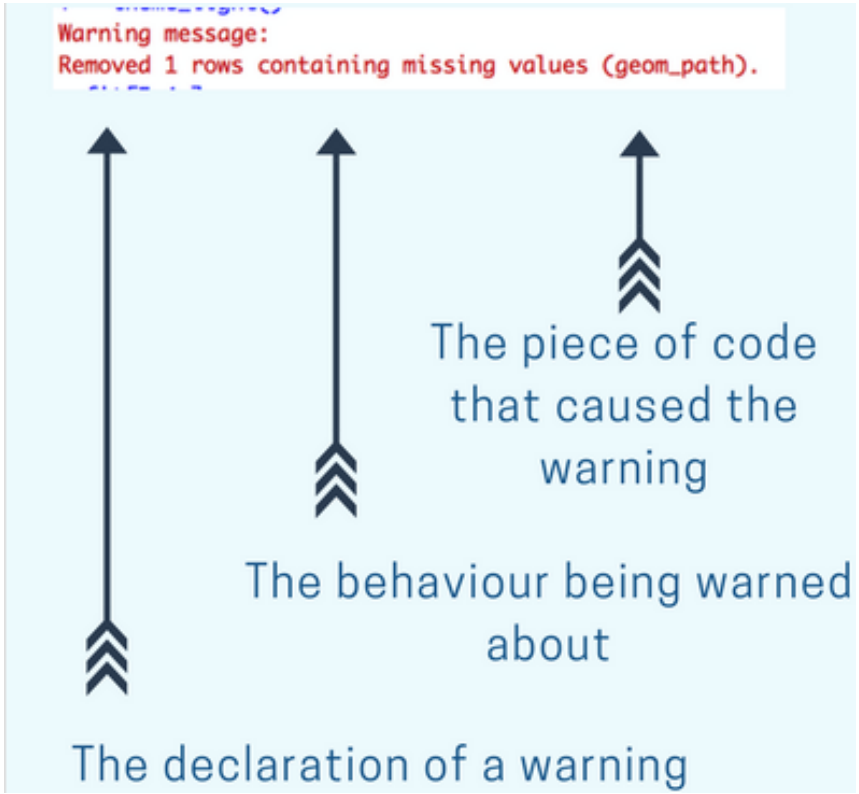
`Error`

`R`: “I can’t and won’t. You won’t force me. I quit. I am done.”



`Warning`

`R`: “I can, but you might not like the output better look at that, my guy.”



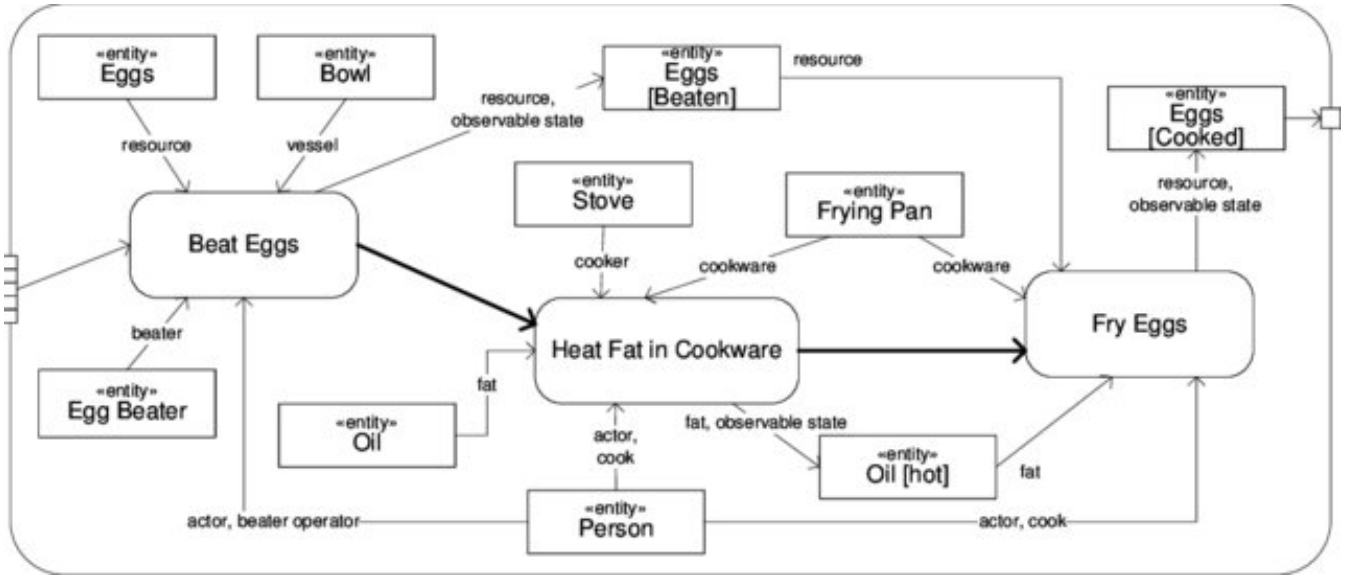
http://rex-analytics.com/decoding-error-messages-r/

Breaking problems down

- Investigate objects and their contents
- Read function help files (especially the arguments and values sections)
- Simulate data if in doubt

Breaking problems down

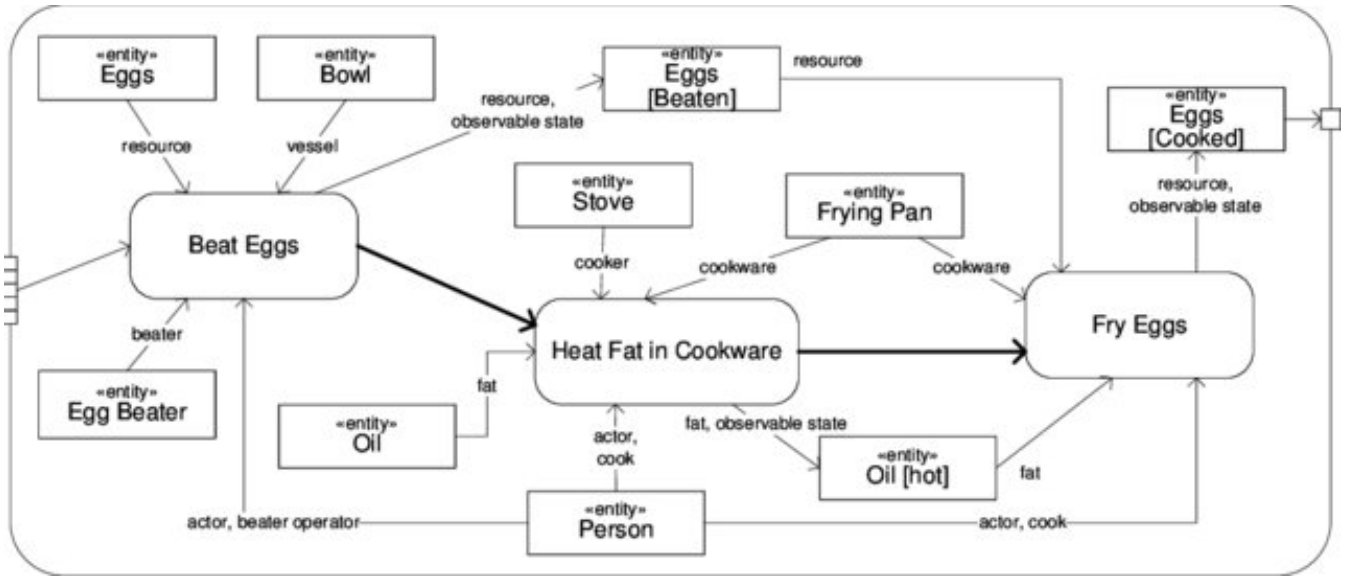
- Investigate objects and their contents
- Read function help files (especially the arguments and values sections)
- Simulate data if in doubt



Sousa et al., 2007

Breaking problems down

- Investigate objects and their contents
- Read function help files (especially the arguments and values sections)
- Simulate data if in doubt



Sousa et al., 2007

Your analyses hardly ever fail at all steps.

What it is and why you need to care

Reproducibility ensures:

- Honest and true reporting
- Results can be scrutinised
- Problems can be traced and identified
- Co-authors and reviewers get less annoyed

Our Goal:

An analysis is **reproducible** when it can be executed by different researchers across operating systems with a **single button prompt** and return the same results for everyone.

Coding Styles

- **Code**
  - *Consistency*
    - The same result on every execution.
    - The same naming scheme throughout.
  - *Self-contained* - Code runs in an empty `R` environment to completion.
- **Documentation**
  - *Commenting* - Informative comments for:
    - What each line is doing/supposed to do.
    - Why each line is doing what it does/is supposed to do.
  - *Header* - Information of what the script does, and who authored it.
  - *File-Versioning* - GitHub or an equivalent to trace when a script broke.

Packages

Never include forced **package installation** in a script.

Instead, do something like this:

```
install.load.package <- function(x) {
  if (!require(x, character.only = TRUE))
    install.packages(x, repos='http://cran.us.r-project.org')
  require(x, character.only = TRUE)
}
package_vec <- c(
  "rethinking", # for quadratic approximation of posteriors
  "reshape2" # for data reformatting
)
sapply(package_vec, install.load.package)
```

Working Directories

Never include **hard-coded** directory indexing.

Make an Omlette out of Easter Eggs

1. Find the egg(s)/data
2. Open the egg(s)/data
3. Make the egg(s) into an omelette / do your analysis

.... but... where are the eggs/the data?

Working Directories

Never include **hard-coded** directory indexing.

Make an Omlette out of Easter Eggs

1. Find the egg(s)/data
2. Open the egg(s)/data
3. Make the egg(s) into an omelette / do your analysis

.... but... where are the eggs/the data?

Working Directories

Never include **hard-coded** directory indexing.

Make an Omlette out of Easter Eggs

1. Find the egg(s)/data
2. Open the egg(s)/data
3. Make the egg(s) into an omelette / do your analysis

.... but... where are the eggs/the data?

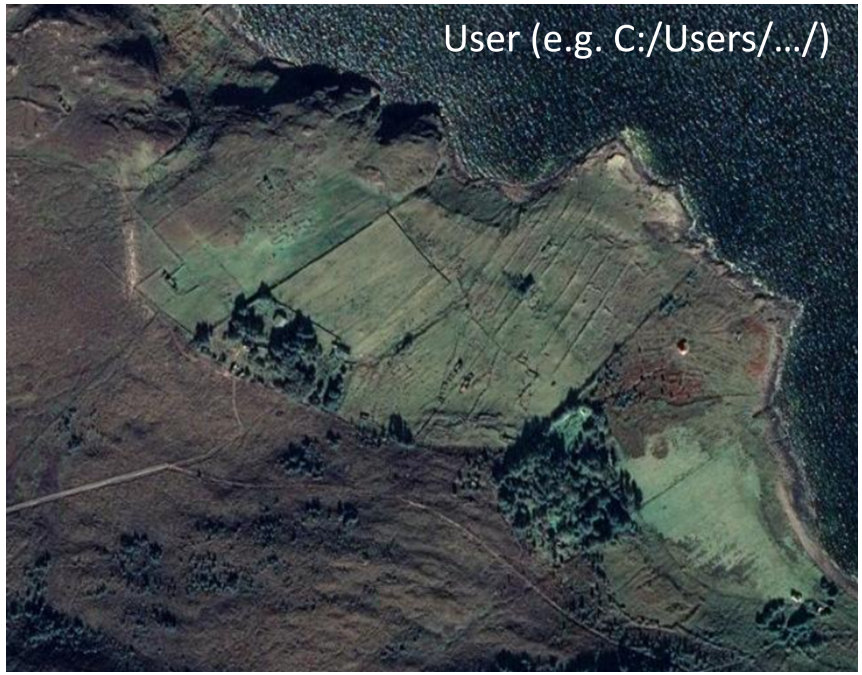
Working Directories

Never include **hard-coded** directory indexing.

Make an Omlette out of Easter Eggs

1. Find the egg(s)/data
2. Open the egg(s)/data
3. Make the egg(s) into an omelette / do your analysis

.... but... where are the eggs/the data?



Working Directories

Never include **hard-coded** directory indexing.

Make an Omlette out of Easter Eggs

1. Find the egg(s)/data
2. Open the egg(s)/data
3. Make the egg(s) into an omelette / do your analysis

.... but... where are the eggs/the data?

Working Directories

Never include **hard-coded** directory indexing.

Make an Omlette out of Easter Eggs

1. Find the egg(s)/data
2. Open the egg(s)/data
3. Make the egg(s) into an omelette / do your analysis

.... but... where are the eggs/the data?

But what if you want to hunt for Easter Eggs/data somewhere other than the garden at Achmore/your specific project folder?

Working Directories

Never include **hard-coded** directory indexing.

Soft-code your working directories from a base/project directory:

```
Dir.Base <- getwd() # to find the project folder
Dir.Data <- file.path(Dir.Base, "Data") # index the data folder
```

Now we have our directories indexed without any hardcoding:

```
Dir.Base
```

```
## [1] "D:/Documents/Teaching/Excursions-into-Biostatistics/Troubleshooting - Isolating Issues and Asking Questions"
```

```
Dir.Data
```

```
## [1] "D:/Documents/Teaching/Excursions-into-Biostatistics/Troubleshooting - Isolating Issues and Asking Questions/Da
```

Randomness

Always make **random processes reproducible**.

Imagine you survey the elevation of a random surface with paratroopers:

- Clouds obscure the landscape
- All flights are on the same route
- Troopers report elevation of their landing
- There is no drift of paratroopers
- Troopers jump at fixed intervals starting at a certain time after take-off

Randomness

Always make **random processes reproducible**.

Imagine you survey the elevation of a random surface with paratroopers:

- Clouds obscure the landscape
- All flights are on the same route
- Troopers report elevation of their landing
- There is no drift of paratroopers
- Troopers jump at fixed intervals starting at a certain time after take-off

Randomness

Always make **random processes reproducible**.

Imagine you survey the elevation of a random surface with paratroopers:

- Clouds obscure the landscape
- All flights are on the same route
- Troopers report elevation of their landing
- There is no drift of paratroopers
- Troopers jump at fixed intervals starting at a certain time after take-off

$\rightarrow$ How can we get the same elevations from the troopers of two different flights?

Randomness

Always make **random processes reproducible**.

Imagine you survey the elevation of a random surface with paratroopers:

- Clouds obscure the landscape
- All flights are on the same route
- Troopers report elevation of their landing
- There is no drift of paratroopers
- Troopers jump at fixed intervals starting at a certain time after take-off

$\rightarrow$ We let all flights commence their jump at the same time after take-off.

$\rightarrow$ How can we get the same elevations from the troopers of two different flights?

Randomness

Always make **random processes reproducible**.

```
rnorm(5, 1, 0.2)
```

```
## [1] 0.8796 1.1960 1.2139 0.9883 0.9878
```

```
rnorm(5, 1, 0.2)
```

```
## [1] 1.1414 0.7078 1.3137 1.1270 1.0204
```

```
set.seed(42); rnorm(5, 1, 0.2)
```

```
## [1] 1.2742 0.8871 1.0726 1.1266 1.0809
```

```
set.seed(42); rnorm(5, 1, 0.2)
```

```
## [1] 1.2742 0.8871 1.0726 1.1266 1.0809
```

A seed needs to be set before **every** random process.

What is a Minimal Working Example (MWE)?

A reproducible, simplified version of code producing a desired outcome.

- **Minimal**. Information and code is reduced as much as possible. This involves reducing:
  - Lines of `R` code
  - Number of `R` packages needed
  - Reliance on directory structures
  - Description of work, result, aim, and potential error
- **Complete**. All information, data, and code necessary to execute the MWE is presented.
- **Reproducible**. The MWE returns the same output on all operating systems when executed by any person. Ideally, this happens at the push of a single button.

Minimal Description

The Goal: *I want to use a k-means clustering approach to identify biome classes across Alaska during the year 1982 using GIMMs NDVI 3g annual mean and seasonality values.*

Method(s) & Material(s):

- *Data:*
- GIMMs NDVI 3g (9x9km index of vegetation performance)
- Natural Vector Shapefiles (shapefile-collection of states and provinces)
- *Method:*
- K-Means-Clustering implemented via the `mclust` package

The Problem: Call to `mclustBIC()` produces the following error:

```
Error in mvnXII(data = data, prior = prior) : NA/NaN/Inf
in foreign function call (arg 1)
```

Minimal Data - Before Reducing to MWE

The Data on our end before handing it off:

Data		
Alaska_Shp	Large SpatialPolygonsDataFrame (642.3 kB)	
All1982_ras	Large RasterStack (279660 elements, 2.3 MB)	
gimms_annual	Large RasterLayer (139830 elements, 1.1 MB)	
gimms_raster	Large RasterBrick (3355920 elements, 26.9 MB)	
gimms_raster_mvc	Large RasterBrick (1677960 elements, 13.4 MB)	
gimms_seasonality	Large RasterLayer (139830 elements, 1.1 MB)	
maxi	Large RasterLayer (139830 elements, 1.1 MB)	
mini	Large RasterLayer (139830 elements, 1.1 MB)	
Shapes	Large SpatialPolygonsDataFrame (4594 elements, 54.1 MB)	
Vals1982_mat	num [1:139830, 1:2] NA NA NA NA NA NA NA NA NA NA ...	
Values		
Dir.Base	"D:/Documents/Teaching/Excursions-into-Biostatistics/Troub…	
Dir.Data	"D:/Documents/Teaching/Excursions-into-Biostatistics/Troub…	
Dir.Plots	"D:/Documents/Teaching/Excursions-into-Biostatistics/Troub…	
gimms_files	chr [1:2] "D:/Documents/Teaching/Excursions-into-Biostatis…	
indices	num [1:24] 1 1 2 2 3 3 4 4 5 5 ...	
Negatives	int [1:102595] 264 266 292 878 898 905 1439 1452 1453 1476…	
Position	1665L	

1982Mean.nc	16/02/2021 15:25	NC File	554 KB
1982Season.nc	16/02/2021 15:25	NC File	554 KB
ndvi3g_geo_v1_1982_0106.nc4	09/03/2020 12:55	NC4 File	437,492 KB
ndvi3g_geo_v1_1982_0712.nc4	09/03/2020 12:47	NC4 File	437,492 KB
ne_10m_admin_1_states_provinces.cpg	09/03/2020 14:05	CPG File	1 KB
ne_10m_admin_1_states_provinces.dbf	09/03/2020 14:05	DBF File	11,327 KB
ne_10m_admin_1_states_provinces.prj	09/03/2020 14:05	PRJ File	1 KB
ne_10m_admin_1_states_provinces.READ...	09/03/2020 14:05	Firefox HTML Doc...	29 KB
ne_10m_admin_1_states_provinces.shp	09/03/2020 14:06	SHP File	20,506 KB
ne_10m_admin_1_states_provinces.shx	09/03/2020 14:06	SHX File	36 KB
ne_10m_admin_1_states_provinces.VERSI...	09/03/2020 14:05	Text Document	1 KB
Shapes.zip	09/03/2020 14:05	WinRAR ZIP archive	14,259 KB

Data needs to be **reduced** to the important parts (`All1982_ras`, in this case).
We can export an `R` environment with the `save.image()` function.

The Data for the MWE that we hand over:

Data		
All1982_ras	Large RasterStack (279660 elements, 2.3 MB)	

Data.rar	16/02/2021 15:18	WinRAR archive	181,813 KB
Workspace.RData	16/02/2021 15:16	R Workspace	294 KB

Minimal Code

Original Script:

```r
### PREAMBLE #### ----------------------------------------------------------
rm(list=ls()) # clearing the entire environment

Dir.Base <- getwd() # identifying the current directory
Dir.Data <- paste(Dir.Base, "Data", sep="/") # generating the folder path for data folder
dir.create(Dir.Data) # creating the data folder
Dir.Plots <- paste(Dir.Base, "Plots", sep="/") # generating the folder path for figures folder
dir.create(Dir.Plots) # creating the figures folder

# install.packages("gimms") # for GIMMS NDVI data download
library(gimms)
# install.packages("raster") # for spatial raster format
library(raster)
# install.packages("sp") # for spatialpolygons format
library(sp)
# install.packages("rgdal") # for shapefiles
library(rgdal)
# install.packages("mclust")
library(mclust)

### DOWNLOADING DATA #### ---------------------------------------------------
### GIMMS DATA
gimms_files <- downloadGimms(x = as.Date("1982-01-01"), # download from January 1982
                             y = as.Date("1982-12-31"), # download to December 1982
                             dsn = Dir.Data, # save downloads in data folder
                             quiet = FALSE # show download progress
)

### SHAPEFILE
download.file("https://www.naturalearthdata.com/http//www.naturalearthdata.com/download/10m/cultural
/ne_10m_admin_1_states_provinces.zip",
              destfile = paste(Dir.Data, "Shapes.zip", sep="/")) # destination file
unzip(paste(Dir.Data, "Shapes.zip", sep="/"), # which file to unzip
      exdir = Dir.Data) # where to unzip to

### READING DATA #### -------------------------------------------------------
### GIMMS DATA
gimms_files <- file.path(Dir.Data, list.files(path = Dir.Data, pattern = "ndvi3g_geo_v1")) # identify the downloaded GIMMS
files
gimms_raster <- rasterizeGimms(x = gimms_files, # the data we rasterize
                               remove_header = TRUE # we don't need the header of the data
)
gimms_raster # some information about the raster stack we created here

### SHAPEFILE
Shapes <- readOGR(Dir.Data, # where to look for the file
                  "ne_10m_admin_1_states_provinces", # the file name
                  verbose = FALSE) # we don't want an overview of the loaded data
Position <- which(Shapes$name_en == "Alaska") # find the english name that's "Alaska" in our shapefiles
Alaska_Shp <- Shapes[Position,] # extract the Alaska shapefile
plot(Alaska_Shp) # plot it for inspection
extent(Alaska_Shp) # extent clearly shows the super-eastern coordinates
Alaska_Shp <- crop(Alaska_Shp, # what to crop
                   extent(-190, -130, 51, 71)) # which extent to crop to
plot(Alaska_Shp) # visualising the cropped product

### HANDLING DATA #### ------------------------------------------------------
### CROP DATA
# Crop
gimms_raster <- crop(gimms_raster, # data to be cropped
                     extent(Alaska_Shp)) # cropped Alaska extent
# Mask (this keeps only pixels that fall into our shapefile)
gimms_raster <- mask(gimms_raster, # cropped data
                     Alaska_Shp) # cropped Alaska shapefile

### MONTHLY AGGREGATES
indices <- monthlyIndices(gimms_files) # generate month indices from the data
gimms_raster_mvc <- monthlyComposite(gimms_raster, # the data
                                     indices = indices # the indices
                                     )
gimms_raster_mvc # some information about our monthly composite raster stack
Negatives <- which(values(gimms_raster_mvc) < 0) # identify all negative values
values(gimms_raster_mvc)[Negatives] <- 0 # set threshold for barren land (NDVI<0)
gimms_raster_mvc
plot(gimms_raster_mvc)

### ANNUAL AGGREGATES
```

MWE Script:

```r
### PREAMBLE #### ----------------------------------------------------------
rm(list=ls()) # clearing the entire environment

install.load.package <- function(x) {
  if (!require(x, character.only = TRUE))
    install.packages(x, repos='http://cran.us.r-project.org')
  require(x, character.only = TRUE)
}
package_vec <- c(
  "raster", # for handling raster format
  "mclust" # for k-means clustering
)
sapply(package_vec, install.load.package)

### LOADING DATA #### -------------------------------------------------------
load("Workspace.RData")

### CLUSTER ANALYSIS #### ---------------------------------------------------

### EXTRACTING DATA
Vals1982_mat <- values(All1982_ras)
rownames(Vals1982_mat) <- 1:dim(Vals1982_mat)[1] # rownames to index raster cell number
```

We removed code for:

- Download of data
- Cropping and masking of data
- Calculation of compound metrics

Complete Reporting

We also supply:

- Archive of our raw data directory
- Script which contains all the code we removed from full script when creating MWE script

We may also want to provide:

- `sessionInfo()`
- `rstudioapi::versionInfo()`
- Metadata if needed

MWE Script:

```r
# ############################################################## #
# PROJECT: [BIOME SHIFTS] Shifting gimms across the Arctic as seen from Space
# CONTENTS: Download, and masking of GIMMS NDVI data across Alaska for 1982
# AUTHOR: Erik Kusch
# EDIT: 16/02/21
# ############################################################## #

### PREAMBLE #### ------------------------------------------------------------
rm(list=ls()) # clearing the entire environment

install.load.package <- function(x) {
  if (!require(x, character.only = TRUE))
    install.packages(x, repos='http://cran.us.r-project.org')
  require(x, character.only = TRUE)
}
package_vec <- c(
  "raster", # for handling raster format
  "mclust" # for k-means clustering
)
sapply(package_vec, install.load.package)

### LOADING DATA #### --------------------------------------------------------
### GIMMS DATA
gimms_files <- downloadGimms(x = as.Date("1982-01-01"), # download from January 1982
                             y = as.Date("1982-12-31"), # download to December 1982
                             dsn = Dir.Data, # save downloads in data folder
                             quiet = FALSE # show download progress
)

### SHAPEFILE
download.file("https://www.naturalearthdata.com/http//www.naturalearthdata.com/download/10m/cultural
/ne_10m_admin_1_states_provinces.zip",
              destfile = paste(Dir.Data, "Shapes.zip", sep="/")) # destination file
unzip(paste(Dir.Data, "Shapes.zip", sep="/"), # which file to unzip
      exdir = Dir.Data) # where to unzip to

### READING DATA #### --------------------------------------------------------
### GIMMS DATA
gimms_files <- file.path(Dir.Data, list.files(path = Dir.Data, pattern = "ndvi3g_geo_v1")) # identify the downloaded GIMMS
files
gimms_raster <- rasterizeGimms(x = gimms_files, # the data we rasterize
                               remove_header = TRUE # we don't need the header of the data
)
gimms_raster # some information about the raster stack we created here

### SHAPEFILE
Shapes <- readOGR(Dir.Data, # where to look for the file
                  "ne_10m_admin_1_states_provinces", # the file name
                  verbose = FALSE) # we don't want an overview of the loaded data
Position <- which(Shapes$name_en == "Alaska") # find the english name that's "Alaska" in our shapefiles
Alaska_Shp <- Shapes[Position,] # extract the Alaska shapefile
plot(Alaska_Shp) # plot it for inspection
extent(Alaska_Shp) # extent clearly shows the super-eastern coordinates
Alaska_Shp <- crop(Alaska_Shp, # what to crop
                   extent(-190, -130, 51, 71)) # which extent to crop to
plot(Alaska_Shp) # visualising the cropped product

### HANDLING DATA #### -------------------------------------------------------
### CROP DATA
# Crop
gimms_raster <- crop(gimms_raster, # data to be cropped
                     extent(Alaska_Shp)) # cropped Alaska extent
# Mask (this keeps only cells that fall into our shapefile)
gimms_raster <- mask(gimms_raster, # cropped data
                     Alaska_Shp) # cropped Alaska shapefile

### MONTHLY AGGREGATES
indices <- monthlyIndices(gimms_files) # generate month indices from the data
gimms_raster_mvc <- monthlyComposite(gimms_raster, # the data
                                     indices = indices # the indices
)
gimms_raster_mvc # some information about our monthly composite raster stack
Negatives <- which(values(gimms_raster_mvc) < 0) # identify all negative values
values(gimms_raster_mvc)[Negatives] <- 0 # set threshold for barren land (NDVI<0)
gimms_raster_mvc
plot(gimms_raster_mvc)

### ANNUAL AGGREGATES
gimms_annual <- calc(gimms_raster_mvc, # data from which to calculate
```

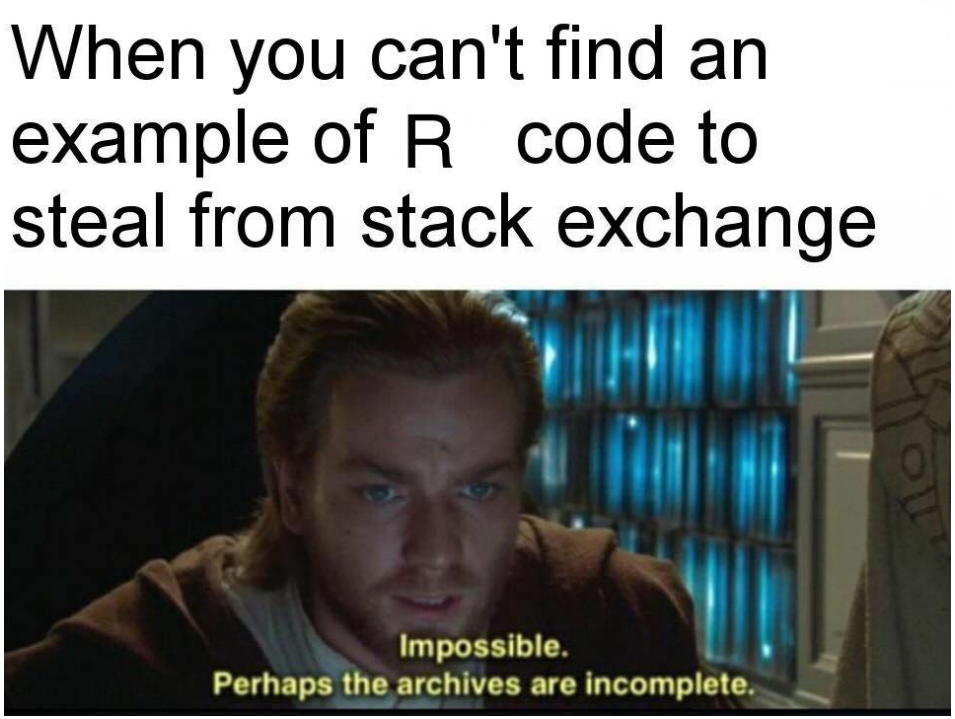
Reproducible Issues

We test our MWE script:

- In a new, empty directory by sourcing the script
- For completeness by running it on an empty `R` environment and by adding `rm(list=ls())` to the start of the script

Our script still produces the same error?

$\rightarrow$ We ship it off to someone whom we hope can help us.



https://knowyourmeme.com/photos/1297214-impossible-perhaps-the-archives-are-incomplete