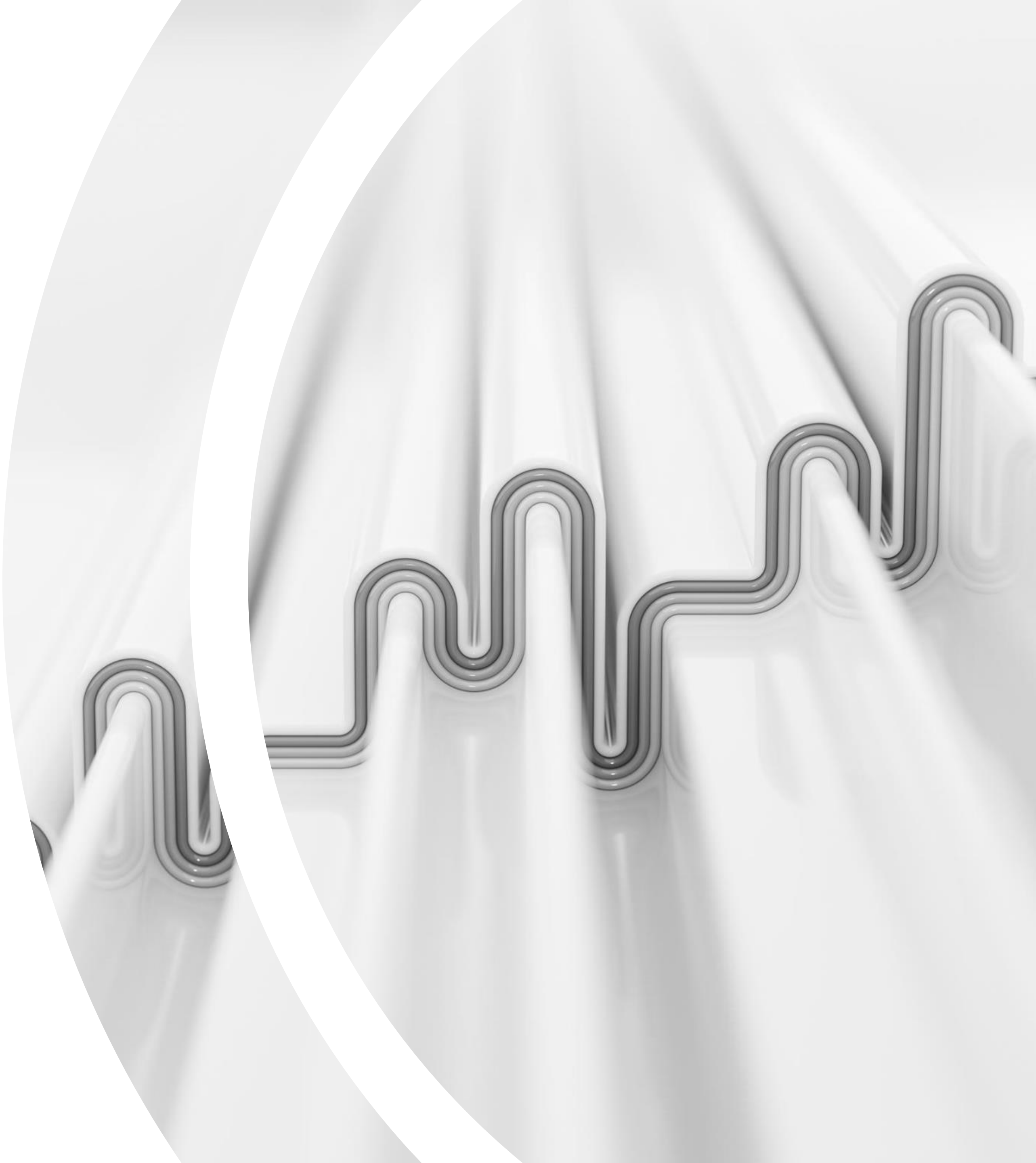
Developing a research pipeline

Esther Jones
Biomathematics & Statistics Scotland

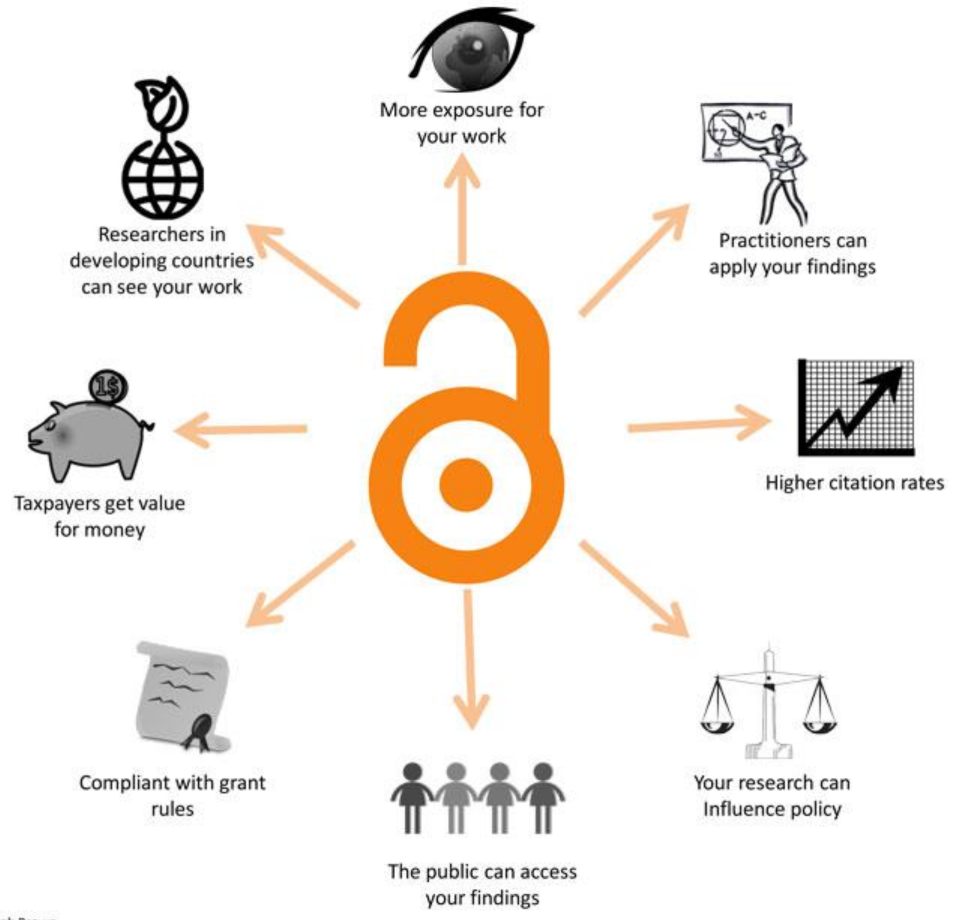


Open Science

Open science is a movement to encourage the free exchange of knowledge and resources to wide access

It aims to make research more transparent, accessible, and collaborative.

Sharing research data, methods, and results openly and freely with the scientific community and the public


More exposure for your work
Practitioners can apply your findings
Higher citation rates
Your research can Influence policy
The public can access your findings
Compliant with grant rules
Taxpayers get value for money
Researchers in developing countries can see your work

https://www.jisc.ac.uk/guides/an-introduction-to-open-access

Open access

Requirement for published articles from funders

Green (self-archiving after embargo) or gold (immediate access - APC)

Pre-print servers

Data are often required to be published with a DOI

Data analysis code and software tools may be required to be shared

Open access

Requirement for published articles from funders

Green (self-archiving after embargo) or gold (immediate access - APC)

Pre-print servers

Data are often required to be published with a DOI

Data analysis code and software tools may be required to be shared

What’s a research analysis pipeline?

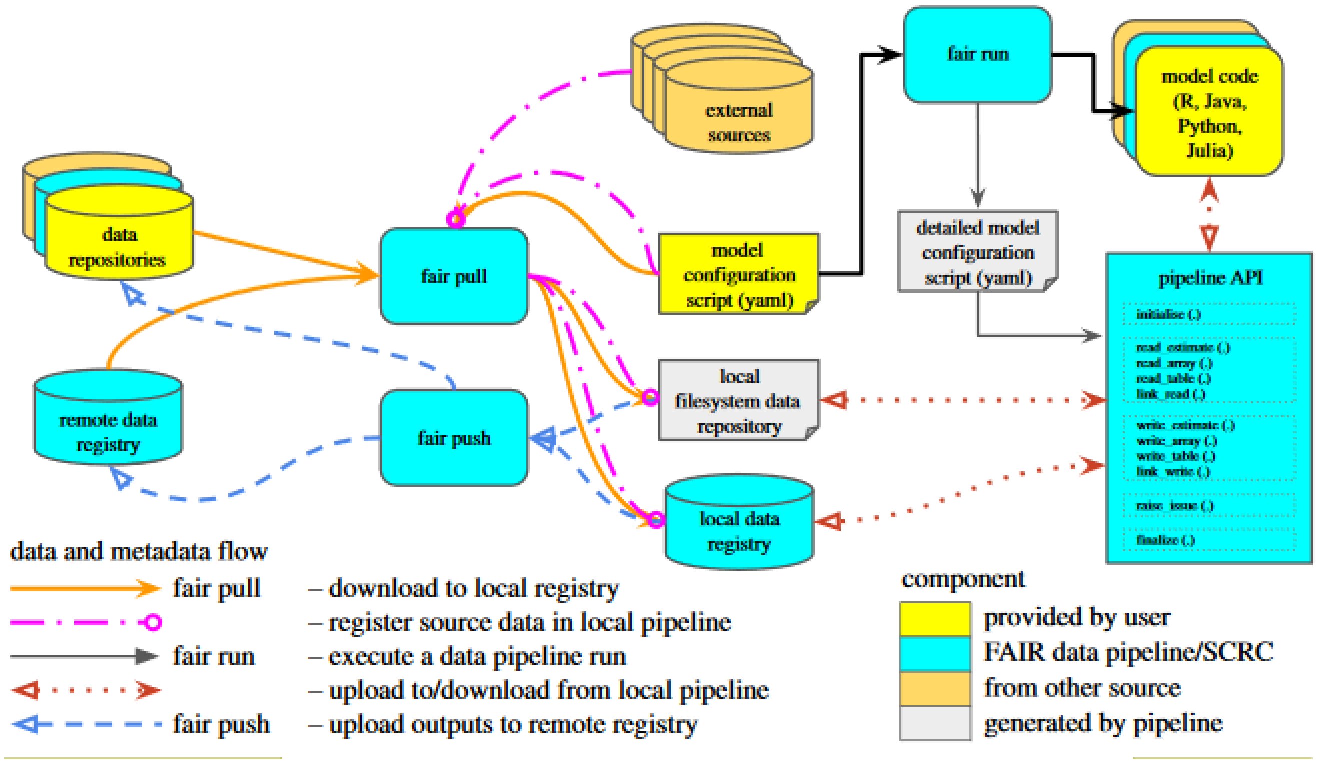
- A systematic and structured approach to undertaking and recording the analytical components of research
- Can include importing data, cleaning & exploratory analysis, modelling, results, outputs, tools, reports, and manuscripts
- Can be extended to collecting, storing, and archiving data
- Using existing frameworks or developing a pipeline can make open access requirements easier to achieve, and promotes open science goals

Why develop a research pipeline?

- Efficiency
- Reproducibility
- Transparency
- Collaboration
- Automation
- Scalability

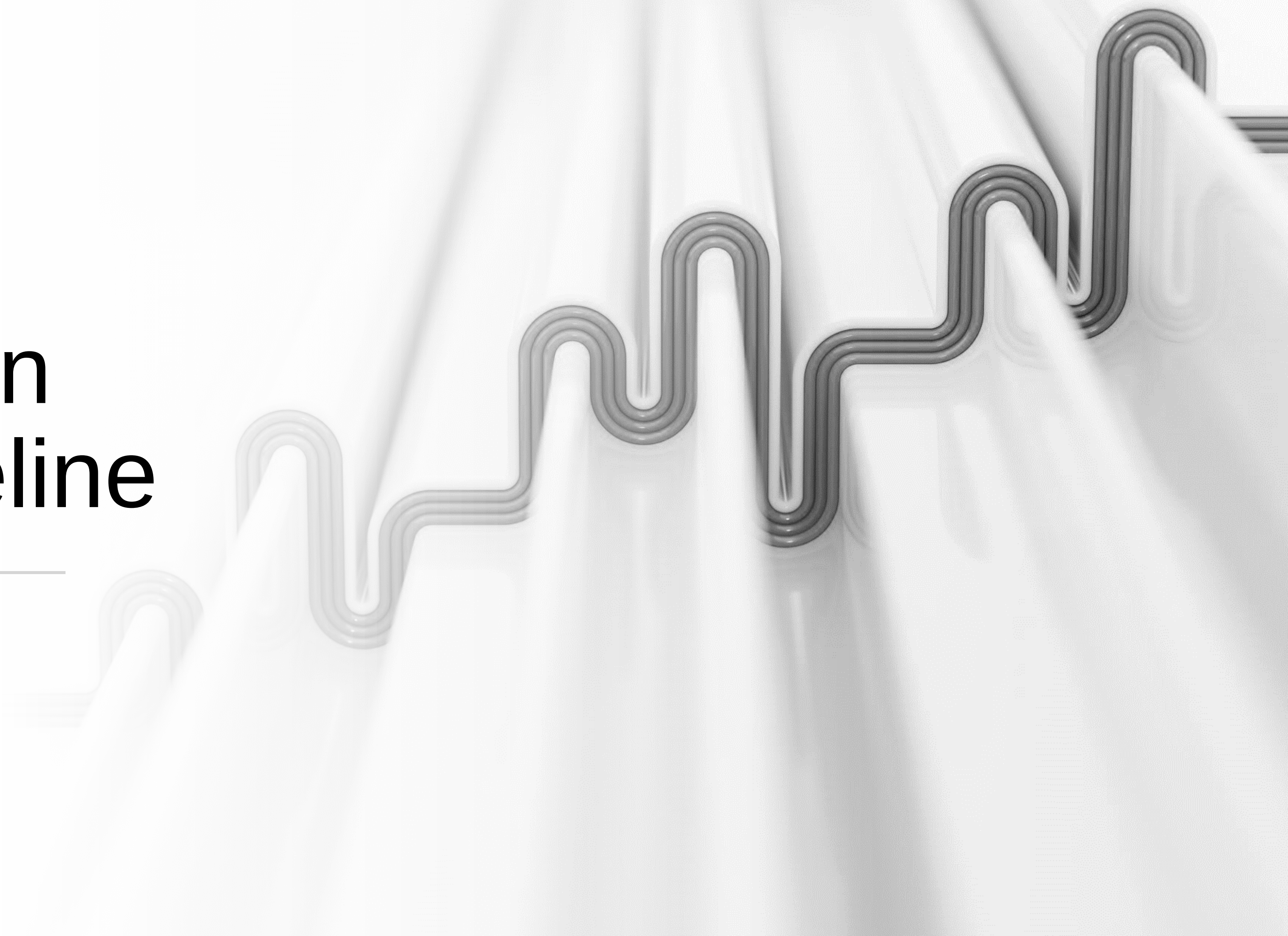
FAIR data pipeline

- Findable, Interoperable and Reusable (FAIR) (www.fairdatapipeline.org)
- Allows data to be tracked and used in epidemiological modelling
- APIs are written in C++, Java, Julia, Python, and R and can be called to incorporate data into modelling
- Relational database with metadata & local filesystem downloading/uploading to remote registry and data store



https://doi.org/10.1098/rsta.2021.0300

Case study: developing an analysis pipeline



Offshore Renewables Group

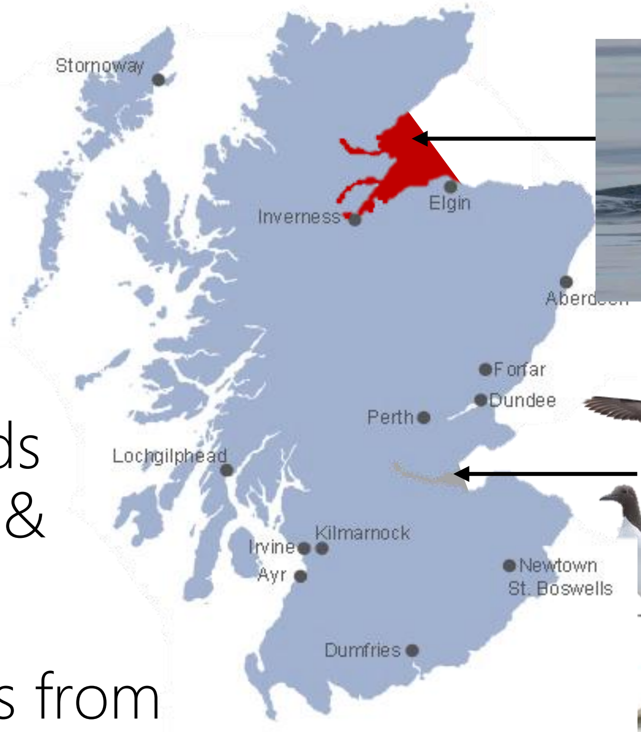
Short term industry funded projects → long term UKRI research

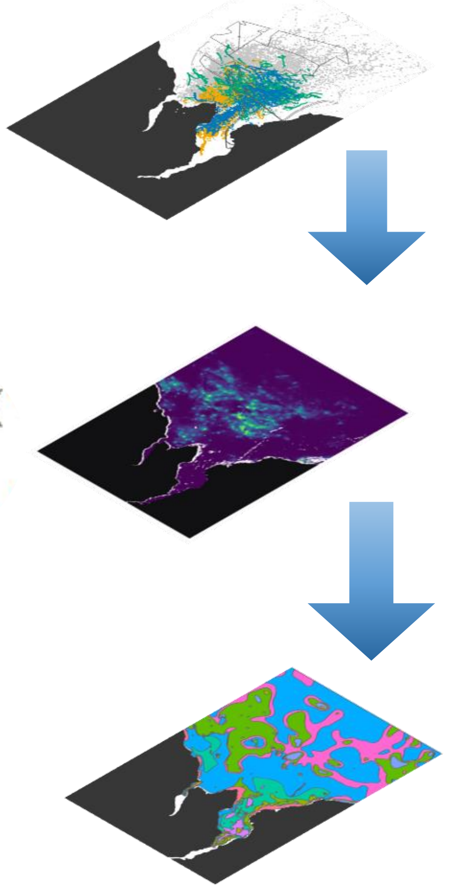
…but they usually have similar components

- Data
- Exploratory analysis
- Modelling
- Outputs
- Software tool development
- Report/article
- Collaboration between many organisations

PrePARED

Predators + Prey Around Renewable Energy Developments





£5M collaborative project (+ in-kind)

Contemporaneous data on fish, seabirds and marine mammals in the Forth-Tay & Moray Firth in 2022-24

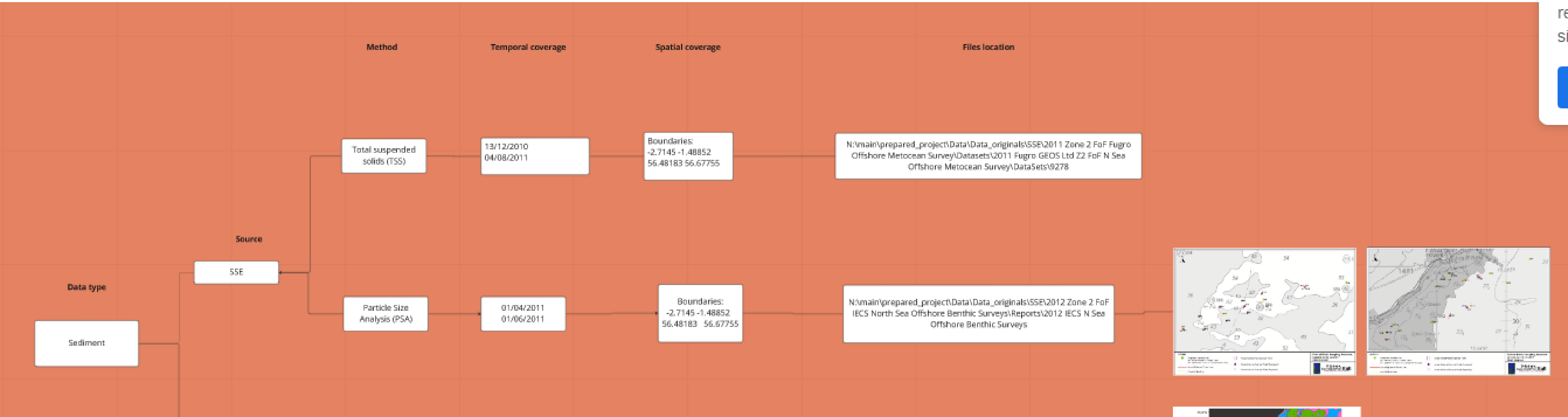
Provides insights into cumulative effects from large scale development for key species

BioSS is leading the seabird-prey modelling to investigate how predator-prey relationships are impacted by offshore renewable developments

Evaluate the extent to which relationships in the Forth-Tay are transferrable to other regions

Data catalogue

- Large amounts of data from multiple sources over large spatial and temporal scales
- Used by multiple teams/organisations
- Useful for data permissions/agreements and sharing



Sediment

SSE

Total suspended solids (TSS)

13/12/2010
04/08/2011

Boundaries:
-2.7145 -1.48852
56.48183 56.67755

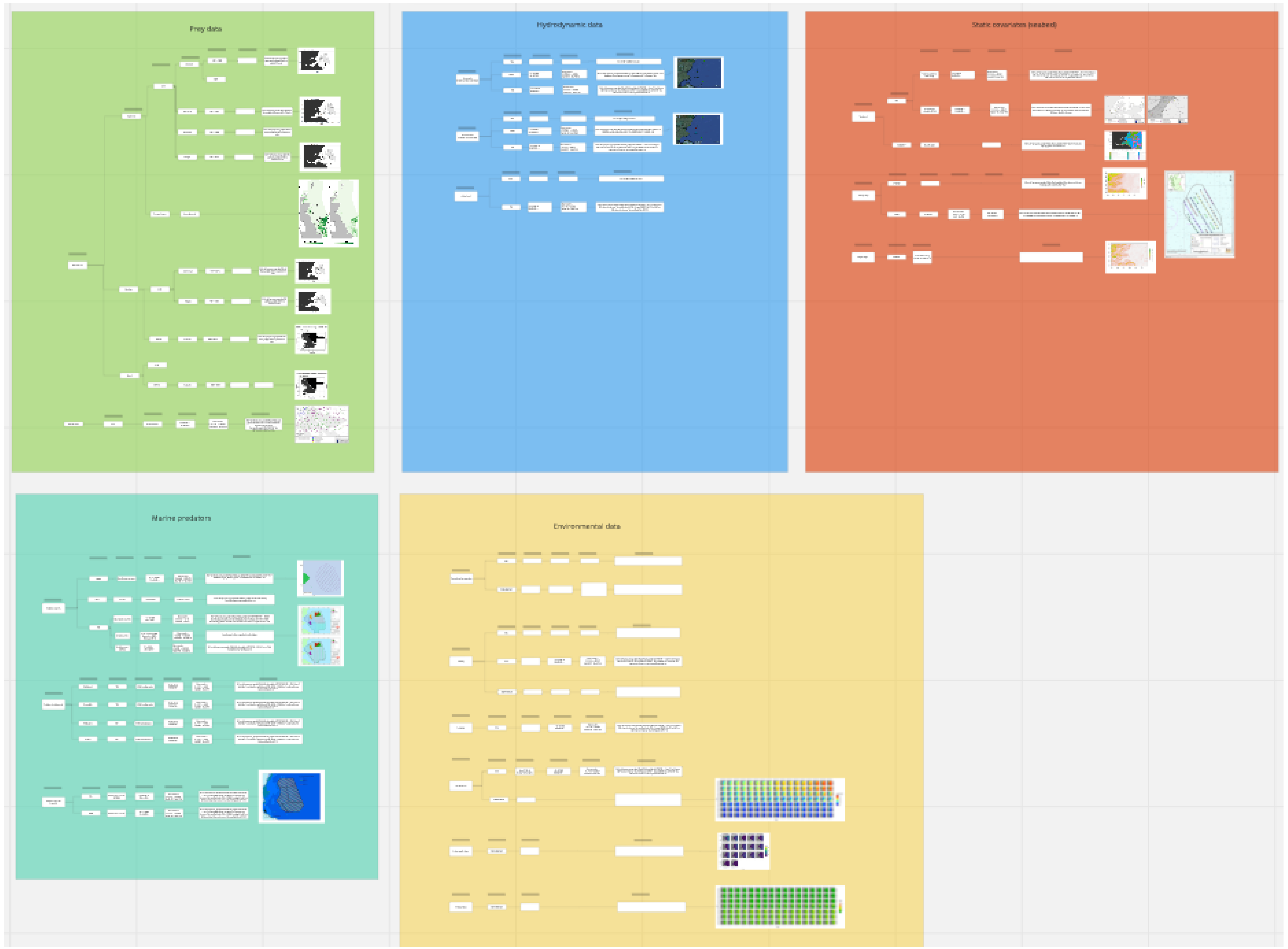
N:\main\prepared_project\Data\Data_originals\SSE\2011 Zone 2 FoF Fugro Offshore Metocean Survey\Datasets\2011 Fugro GEOS Ltd Z2 FoF N Sea Offshore Metocean Survey\DataSets\9278

Particle Size Analysis (PSA)

01/04/2011
01/06/2011

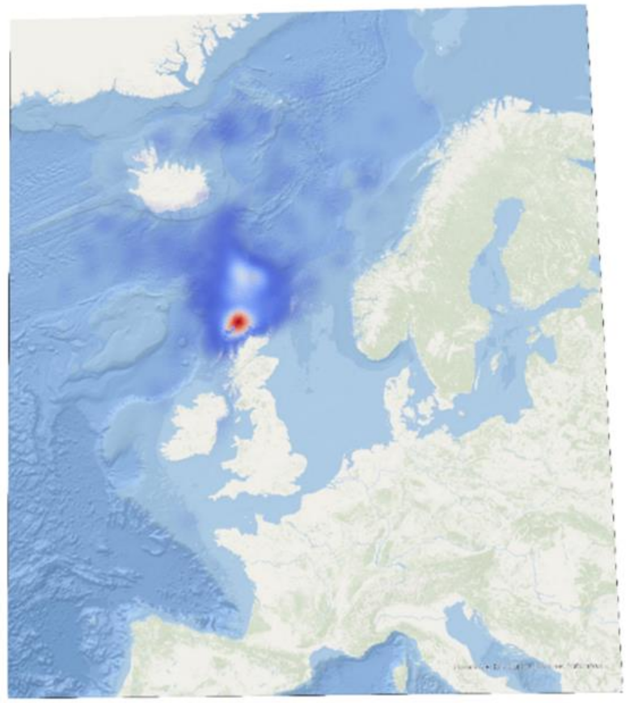
Boundaries:
-2.7145 -1.48852
56.48183 56.67755

N:\main\prepared_project\Data\Data_originals\SSE\2012 Zone 2 FoF IECS North Sea Offshore Benthic Surveys\Reports\2012 IECS N Sea Offshore Benthic Surveys


Prey data
Hydrodynamic data
Static covariates (seabed)
Marine predators
Environmental data

ORJIP Apportioning

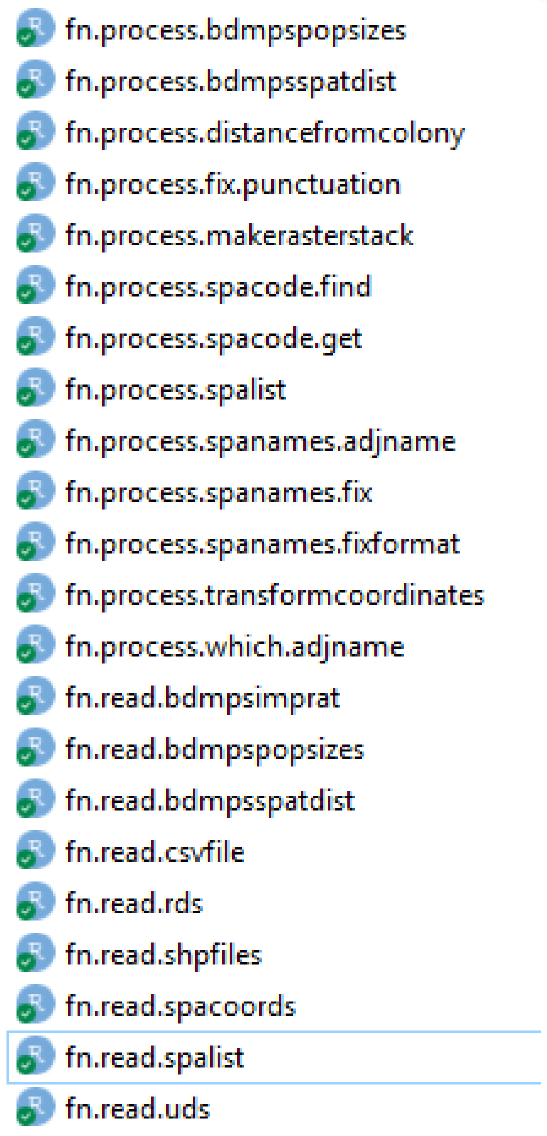
- £100K collaborative project led by UKCEH with BioSS, BTO & MacArthur Green
- Part of project is apportioning of seabirds at-sea in the non-breeding season
- Seabirds are central placed foragers during breeding (summer) but are less constrained in winter
- Developing software tool for apportioning



Writing functions

- When writing code, use functions as much as possible
- It helps make code readable (to other people and future self)
- Functions can often be reused so not reinventing the wheel each time
- Can follow a naming convention to help find functions that may be useful
- Import all functions at once (in R setup)

```
files.sources = list.files(pattern="fn.")
sapply(files.sources, source)
```



```
fn.process.bdmpspopsizes
fn.process.bdmpsspatdist
fn.process.distancefromcolony
fn.process.fix.punctuation
fn.process.makerasterstack
fn.process.spacode.find
fn.process.spacode.get
fn.process.spalist
fn.process.spanames.adjname
fn.process.spanames.fix
fn.process.spanames.fixformat
fn.process.transformcoordinates
fn.process.which.adjname
fn.read.bdmpsimprat
fn.read.bdmpspopsizes
fn.read.bdmpsspatdist
fn.read.csvfile
fn.read.rds
fn.read.shpfiles
fn.read.spacoords
fn.read.spalist
fn.read.uds
```

Documenting functions

- When writing a function, it's useful to explain what it does, input parameters, outputs, and dependencies
- roxygen2 library https://cran.r-project.org/web/packages/roxygen2/vignettes/roxygen2.html
- Standardises documentation
- Can use it to help build a package as it manages NAMESPACE and some of the DESCRIPTION file
- You can add roxygen text to R code manually;
- or automatically using `roxygen2::roxygenise()` to convert roxygen comments to .Rd files

https://roxygen2.r-lib.org/articles/rd.html

Use tags with @ at beginning of the line

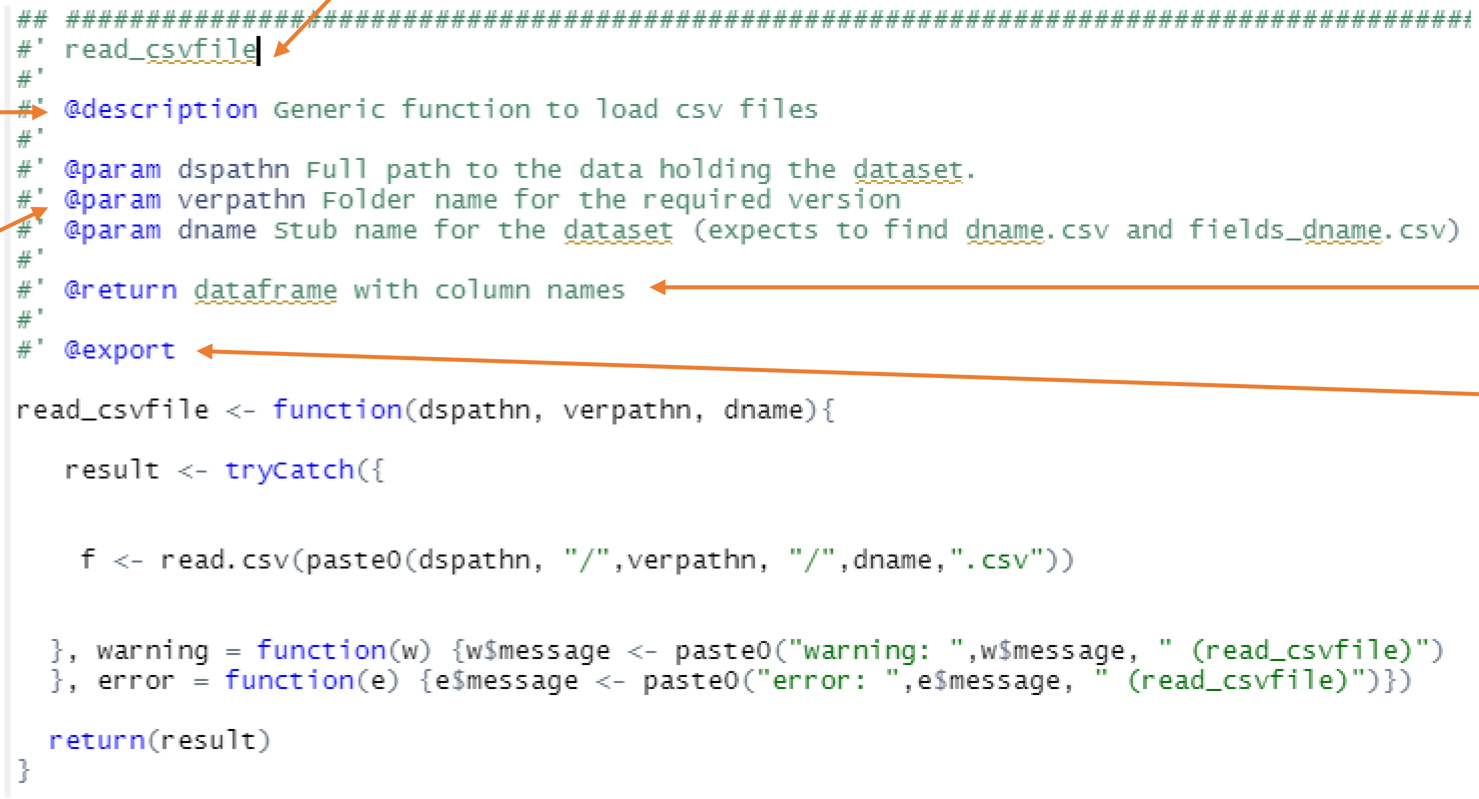
title

Description of function

Name & description of input parameters

Description of output(s)

Use in package



```
## ###########################################################################
#' read_csvfile
#'
#' @description Generic function to load csv files
#'
#' @param dspathn Full path to the data holding the dataset.
#' @param verpathn Folder name for the required version
#' @param dname Stub name for the dataset (expects to find dname.csv and fields_dname.csv)
#'
#' @return dataframe with column names
#'
#' @export

read_csvfile <- function(dspathn, verpathn, dname){

  result <- tryCatch({


   f <- read.csv(paste0(dspathn, "/",verpathn, "/",dname,".csv"))


  }, warning = function(w) {w$message <- paste0("warning: ",w$message, " (read_csvfile)")
  }, error = function(e) {e$message <- paste0("error: ",e$message, " (read_csvfile)")})

  return(result)
}
```

Testing code

Code is often tested informally or on an ad hoc basis

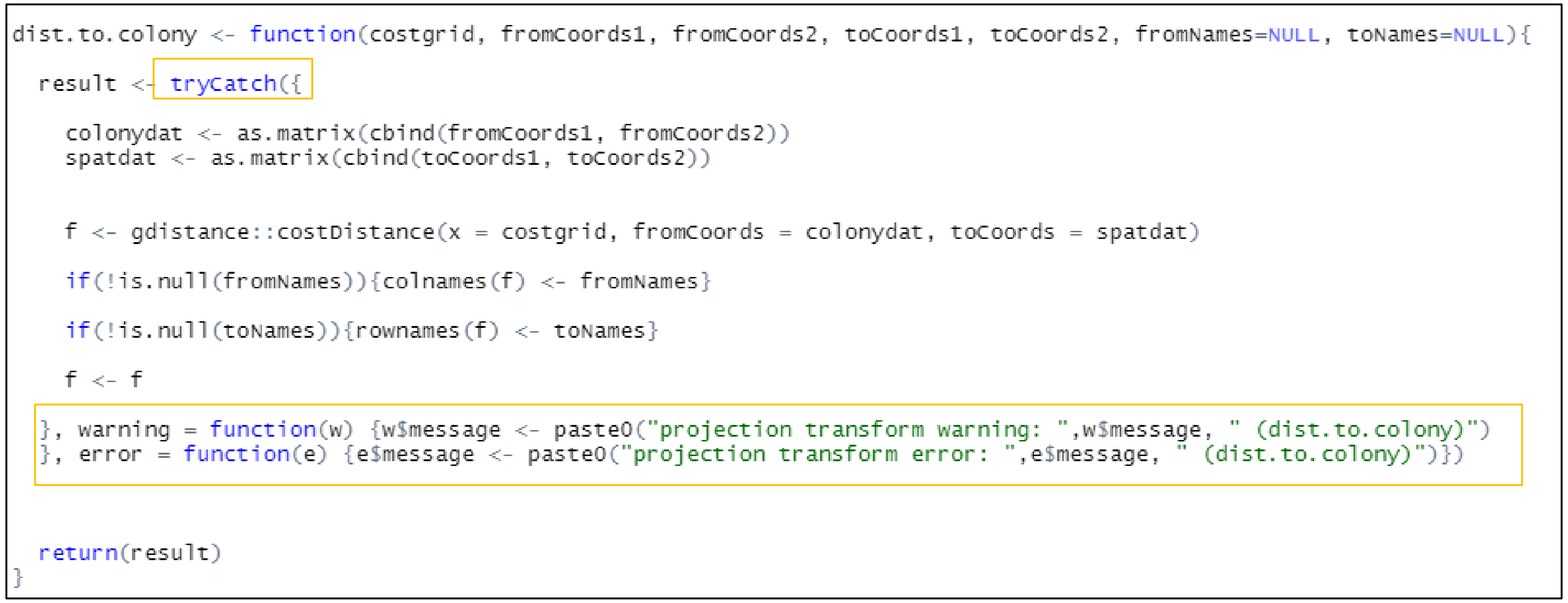
- The outputs looks 'about right'
- Trying different inputs and getting what you expect
- Testing individual functions

However, these tests can get lost and may not have consistency.

Two useful types of testing – **error handling** and **unit testing**

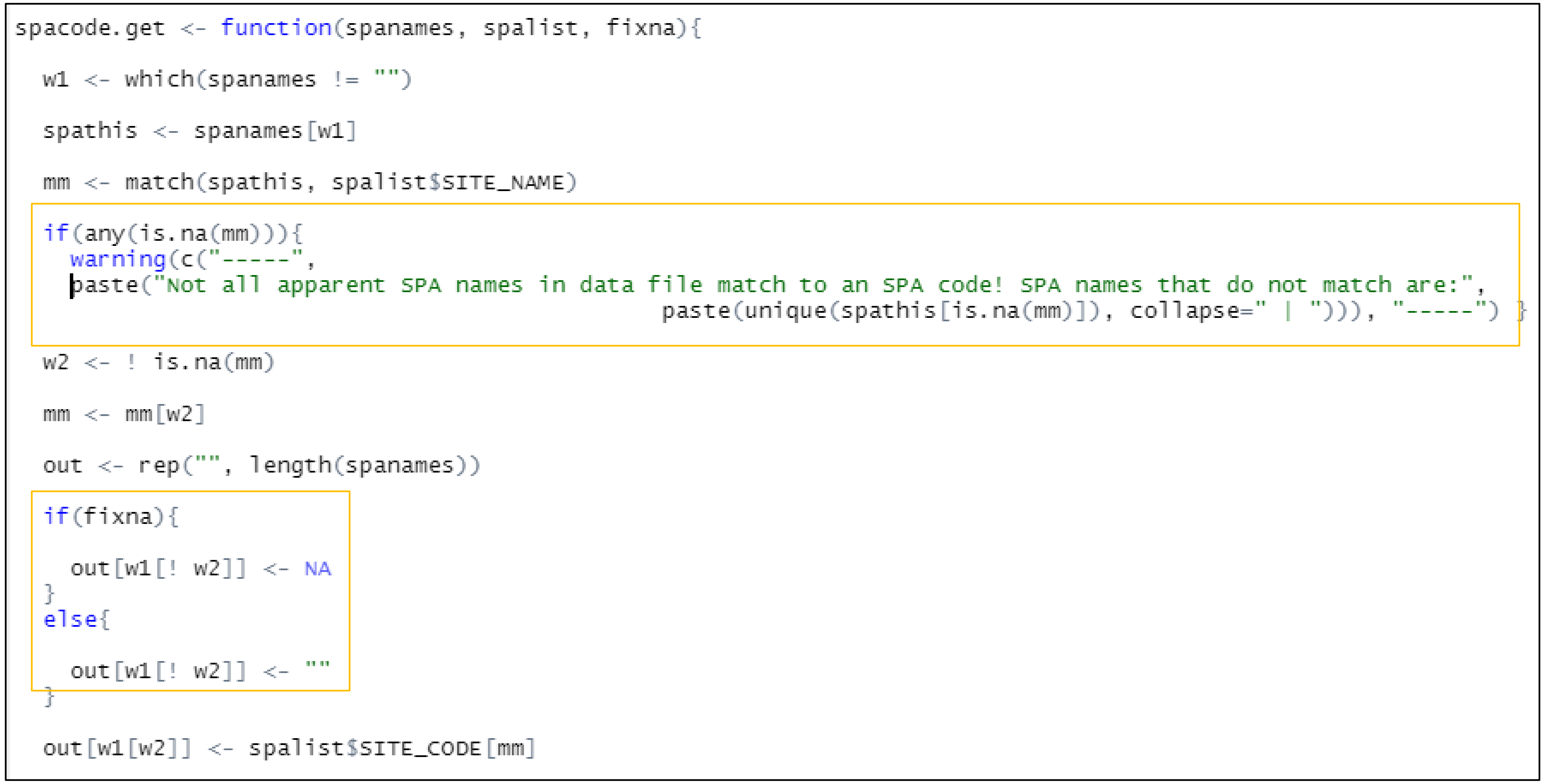
Error handling

- Adding expressions into functions to try and catch common errors
- Can generate informative warning or error messages back to the user



```r
dist.to.colony <- function(costgrid, fromCoords1, fromCoords2, toCoords1, toCoords2, fromNames=NULL, toNames=NULL){

  result <- tryCatch({

    colonydat <- as.matrix(cbind(fromCoords1, fromCoords2))
    spatdat <- as.matrix(cbind(toCoords1, toCoords2))


    f <- gdistance::costDistance(x = costgrid, fromCoords = colonydat, toCoords = spatdat)

    if(!is.null(fromNames)){colnames(f) <- fromNames}

    if(!is.null(toNames)){rownames(f) <- toNames}

    f <- f

  }, warning = function(w) {w$message <- paste0("projection transform warning: ",w$message, " (dist.to.colony)")
  }, error = function(e) {e$message <- paste0("projection transform error: ",e$message, " (dist.to.colony)")})


  return(result)
}
```

Error handling



```
spacode.get <- function(spanames, spalist, fixna){

  w1 <- which(spanames != "")

  spathis <- spanames[w1]

  mm <- match(spathis, spalist$SITE_NAME)

  if(any(is.na(mm))){
    warning(c("-----",
    paste("Not all apparent SPA names in data file match to an SPA code! SPA names that do not match are:",
                                              paste(unique(spathis[is.na(mm)]), collapse=" | "))), "-----") }

  w2 <- ! is.na(mm)

  mm <- mm[w2]

  out <- rep("", length(spanames))

  if(fixna){

    out[w1[! w2]] <- NA
  }
  else{

    out[w1[! w2]] <- ""
  }

  out[w1[w2]] <- spalist$SITE_CODE[mm]
```

Unit testing

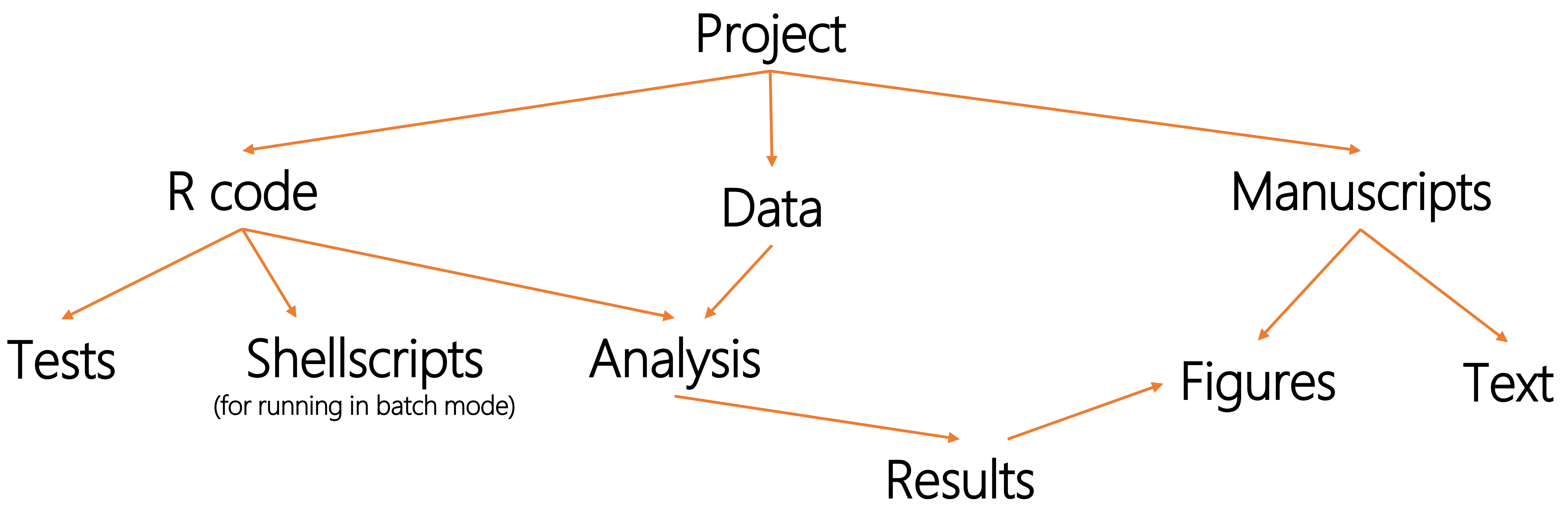
- A structured and automatic way of testing code, from a single function up to an entire R package
- Single tests are written to test one aspect of functionality and then run separately or all together
- Stored and run automatically so if you reorganise or restructure your code, tests will still run
- Makes you think about how you write code and gives confidence that you’ve caught the majority of errors if/when you need to share code – open science

Unit testing

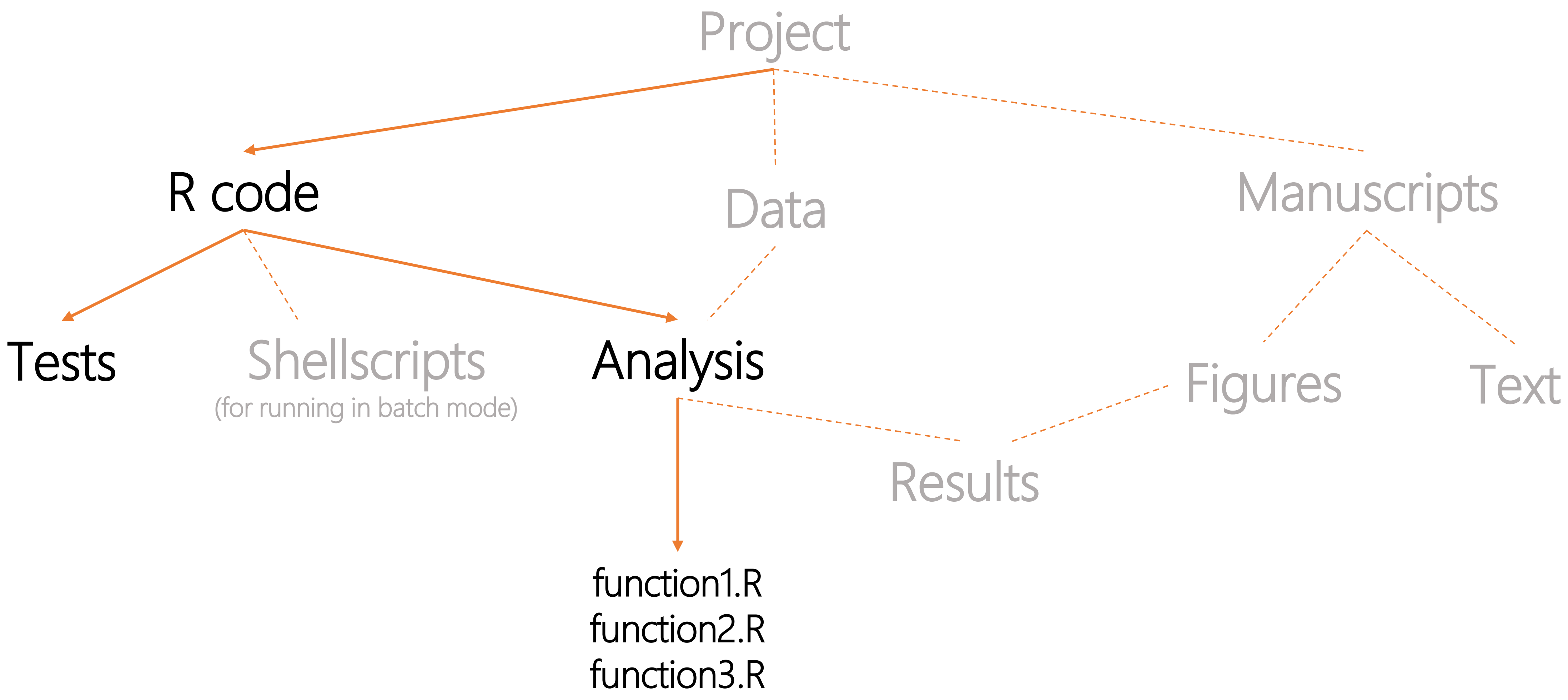
- Guide to testing – ‘R packages’ – Hadley Wickham & Jenny Bryan https://r-pkgs.org/testing-basics.html
- Use a package called ‘testthat’ https://testthat.r-lib.org
- Can use the package ‘usethis’ to help you automatically set up testing https://usethis.r-lib.org/

- You can set up tests/test structure manually or let usethis do it automatically
- At a minimum, set up a directory called **tests/testthat/**

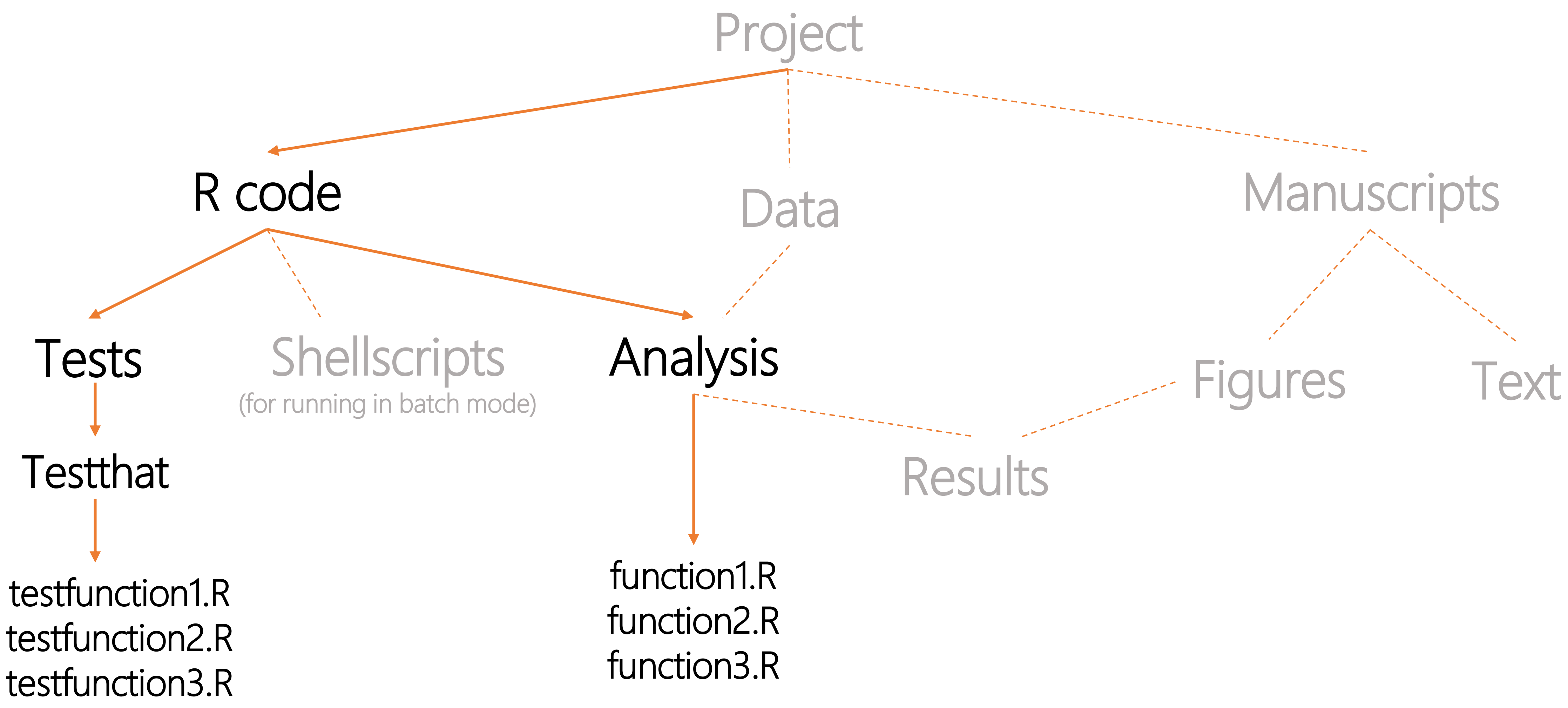
Example of organising a project



Example of organising a project

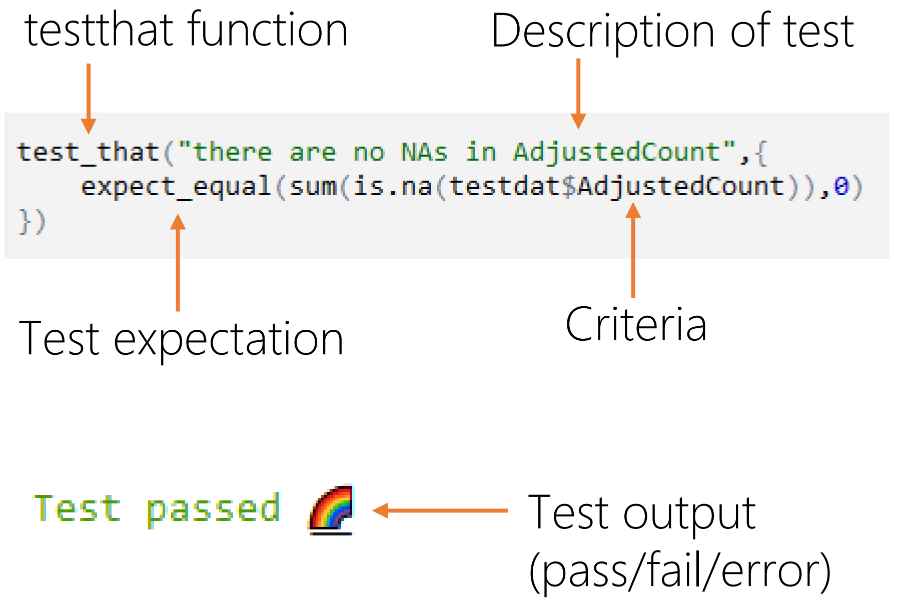


Example of organising a project



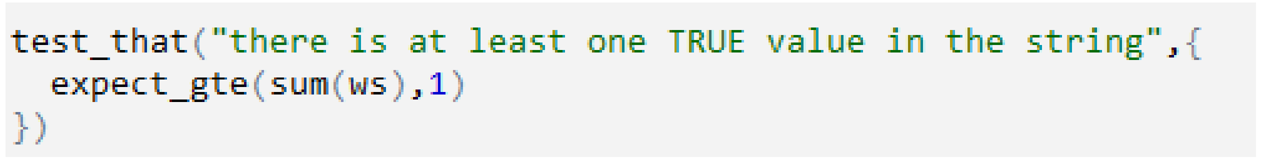
Writing tests

- Keep tests simple - one clause per test
- Write informative descriptions
- You can write as many or as few tests as you want

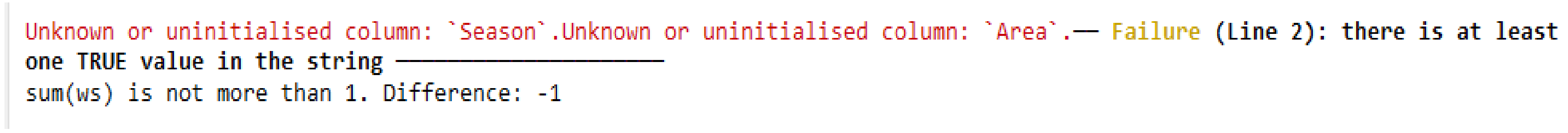


```
test_that("there are no NAs in AdjustedCount",{
    expect_equal(sum(is.na(testdat$AdjustedCount)),0)
})
```

```
Test passed 🌈
```



```
test_that("there is at least one TRUE value in the string",{
  expect_gte(sum(ws),1)
})
```



```
Unknown or uninitialised column: `Season`.Unknown or uninitialised column: `Area`.— Failure (Line 2): there is at least
one TRUE value in the string ——————————————————
sum(ws) is not more than 1. Difference: -1
```

Examples of expectations

https://testthat.r-lib.org/reference/index.html#expectations

Objects	
expect_equal() expect_identical()	does the code return the expected value?
expect_type() expect_s3_class() expect_s4_class()	does the code return an object inheriting from the expected base type, s3 class, or s4 class?
Vectors	
expect_length()	does code return a vector with the specified length?
expect_lt() expect_lte() expect_gt() expect_gte()	does code return a number greater/lesser/equal to expected value?
expect_named()	does the code return a vector with (given) names?
expect_setequal() expect_mapequal()	does code return a vector containing the expected values?
expect_true() expect_false()	does the code return true or false?
expect_vector()	does code return a vector with the expected size?
Side-effects	
expect_error() expect_warning() expect_message() expect_condition()	does code throw an error, warning, message, or other condition?

Functions for running tests

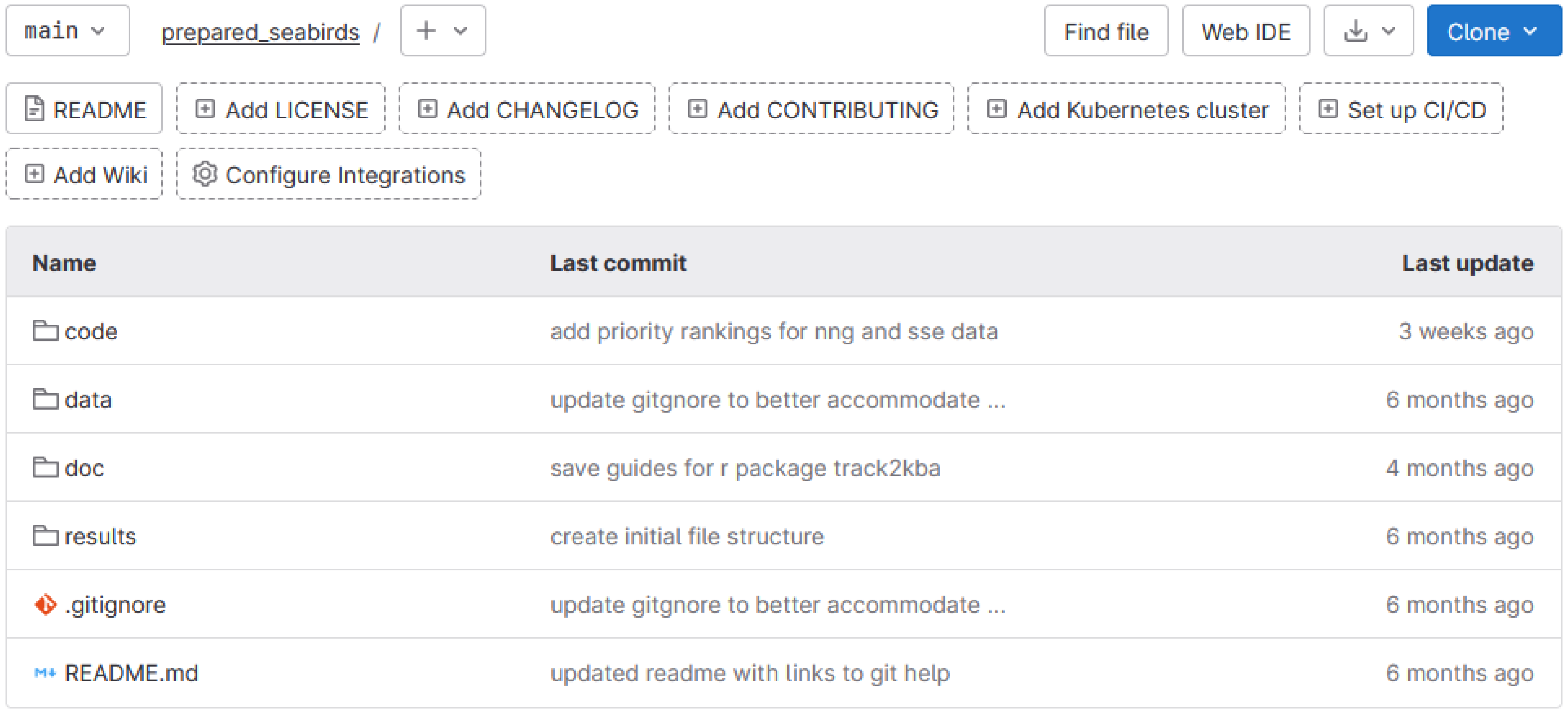
Can run single tests → whole package

https://testthat.r-lib.org/reference/index.html#run-tests

Run tests	
auto_test()	Watches code and tests for changes, rerunning tests as appropriate
auto_test_package()	Watches a packages for changes, rerunning tests as appropriate
test_file()	Runs all tests in a single file
test_package() test check() test_local()	Runs all tests in a package
test_path()	Locate file in testing directory
test_that()	Run a (single) test

Dissemination

- Packaging up into an R library – helpful blog https://godatadriven.com/blog/developing-r-packages-and-data-applications/
- Github



Name	Last commit	Last update
code	add priority rankings for nng and sse data	3 weeks ago
data	update gitgnore to better accommodate ...	6 months ago
doc	save guides for r package track2kba	4 months ago
results	create initial file structure	6 months ago
.gitignore	update gitgnore to better accommodate ...	6 months ago
README.md	updated readme with links to git help	6 months ago

Summary

- Developing or using an existing framework for an analysis pipeline is generally a useful thing to do
- Pipelines can be extensive or you can choose to optimise/focus on one or a few elements (e.g. code testing)
- They are scalable, reusable, and adaptable
- Can really help to achieve open access and open science goals and requirements
- Structured way to develop learning and 'best practice' across an organisation/team

Acknowledgements

- Lee Benson
- Adam Butler
- Sonia Mitchell
- Helen Kettle
- Deena Mobbs (UKCEH)