Darwin Harbour sediment monitoring program analysis application manual

Murray Logan

20/03/2025

Table of contents

About

This document comprises the manual for the Darwin Harbour sediment monitoring program analysis application. It provides information on:

- a broad overview of the structure of the application
- the application dependencies and how to install them
- starting the application
- progressing through the analysis pipeline
- visualising, interpreting and extracting outputs

Structural overview

R Graphical and Statistical Environment offers an ideal platform for developing and running complex statistical analyses as well as presenting the outcomes via professional graphical/tabular representations. As a completely scripted language it also offers the potential for both full transparency and reproducibility. Nevertheless, as the language, and more specifically the extension packages are community developed and maintained, the environment evolves over time. Similarly, the underlying operating systems and programs on which R and its extension packages depend (hereafter referred to as the *operating environment*) also change over time. Consequently, the stability and reproducibility of R codes have a tendency to change over time.

Docker containers

One way to attempt to future proof a codebase that must be run upon a potentially unpredictable operating environment is to **containerise** the operating environment, such that it is preserved to remain unchanged over time. Containers (specifically docker containers) are lightweight abstraction units that encapsulate applications and their dependencies within standardized, self-contained execution environments. Leveraging containerization technology, they package application code, runtime, libraries, and system tools into isolated units (*containers*) that abstract away underlying infrastructure differences, enabling consistent and predictable execution across diverse computing platforms.

Containers offer several advantages, such as efficient resource utilization, rapid deployment, and scalability. They enable developers to build, test, and deploy applications with greater speed and flexibility. Docker containers have become a fundamental building block in modern software development, enabling the development and deployment of applications in a consistent and predictable manner across various environments.

Shiny applications

Shiny is a web application framework for R that enables the creation of interactive and data-driven web applications directly from R scripts. Developed by Rstudio, Shiny simplifies the process of turning analyses into interactive web-based tools without the need for extensive web development expertise.

What makes Shiny particularly valuable is its seamless integration with R, allowing statisticians and data scientists to build and deploy bespoke statistical applications, thereby making data visualization, exploration, and analysis accessible to a broader audience. With its interactive and user-friendly nature, Shiny serves as a powerful tool for sharing insights and engaging stakeholders in a more intuitive and visual manner.

Git and github

Git, a distributed version control system, and GitHub, a web-based platform for hosting and collaborating on Git repositories, play pivotal roles in enhancing reproducibility and transparency in software development. By tracking changes in source code and providing a centralized platform for collaborative work, Git and GitHub enable developers to maintain a detailed history of code alterations. This history serves as a valuable asset for ensuring the reproducibility of software projects, allowing users to trace and replicate specific versions of the codebase.

GitHub Actions (an integrated workflow automation feature of GitHub), automates tasks such as building, testing, and deploying applications and artifacts. Notably, through workflow actions, GitHub Actions can build docker containers and act as a container registry. This integration enhances the overall transparency of software development workflows, making it easier to share, understand, and reproduce projects collaboratively.

Figure 1 provides a schematic overview of the relationship between the code produced by the developer, the Github cloud repositiory and container registry and the shiny docker container run by user.

Installation

Installing docker desktop

To retrieve and run docker containers requires the installation of Docker Desktop on Windows and MacOSx

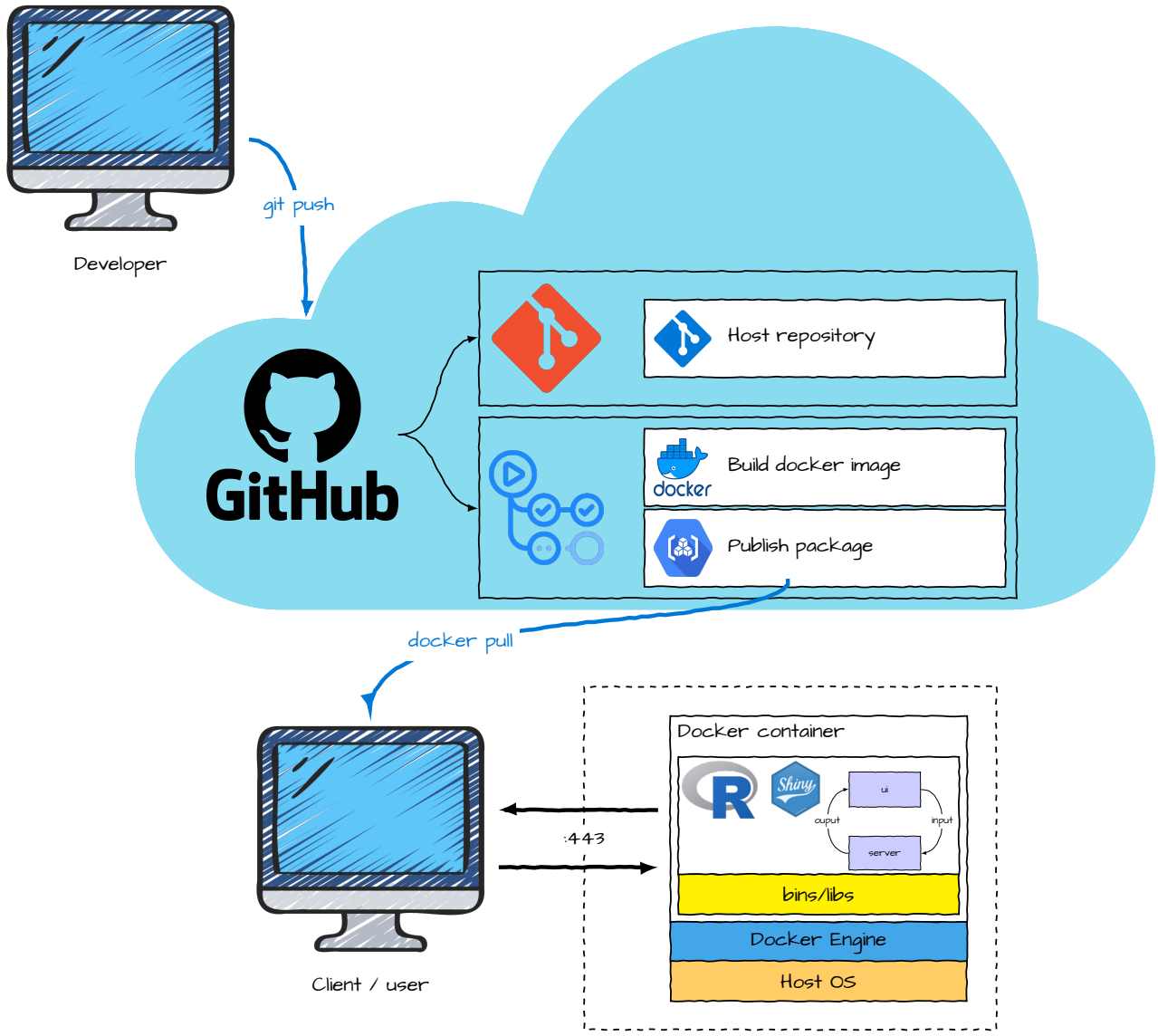


Figure 1: Diagram illustrating the relationship between the code produced by the developer and the shiny docker container utilised by user with a Github cloud conduit. The developed codebase includes a Shiny R application with R backend, `Dockerfile` (instructions used to assemble a full operating environment) and github workflow file (instructions for building and packaging the docker image on github via `actions`).

Windows

The steps for installing Docker Desktop are:

- **Download the Installer:** head to https://docs.docker.com/desktop/install/windows-install/ and follow the instructions for downloading the appropriate installer for your Windows version (Home or Pro).
- **Run the Installer:** double-click the downloaded file and follow the on-screen instructions from the installation wizard. Accept the license agreement and choose your preferred installation location.
- **Configure Resources (Optional):** Docker Desktop might suggest allocating some system resources like CPU and memory. These settings can be adjusted later, so feel free to use the defaults for now.
- **Start the Docker Engine:** once installed, click the "Start Docker Desktop" button. You may see a notification in the taskbar - click it to confirm and allow Docker to run in the background.
- **Verification:** open a terminal (or Powershell) and run `docker --version`. If all went well, you should see information about the installed Docker Engine version.

Additional Tips:

- Ensure Hyper-V (virtualization) is enabled in your BIOS settings for optimal performance.

Installing the and running the app

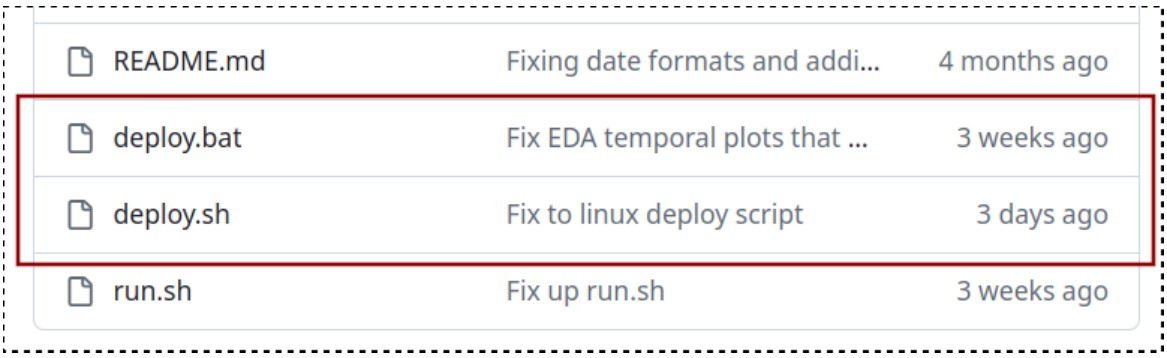
The task of installing and running the app is performed via a single **deploy script** (`deploy.bat` on Windows or `deploy.sh` on Linux/MacOSX/wsl). For this to work properly, the deploy script should be placed in a folder along with a folder (called `input`) that contains the input datasets (in excel format). This structure is illustrated below for Windows.

```
\
|- deploy.bat
|- input
   |- dataset1.xlsx
   |- dataset2.xlsx
```

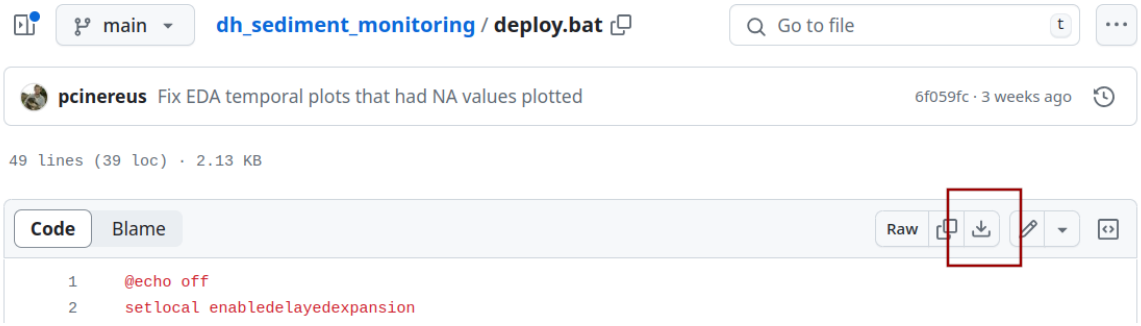
> **Note**
>
> In the above illustration, there are two example datasets (`dataset1.xlsx` and `dataset2.xlsx`). The datasets need NOT be called `dataset1.xlsx`. They can have any name you choose, so long as they are excel files that adhere to the structure outlined in Section .

To set up the above struture:

1. create a new folder on your computer in a location of your choice that you are likely to remember and easily locate (e.g. on the desktop). Whilst the name of the folder is not important, it is recommended that it be named after the project (e.g. `darwin_harbour_sediment_monitoring`).
2. download the deploy script from the projects github repository
   a. go to the projects github repository (https://github.com/open-AIMS/dh_sediment_monitoring.git) in a browser
   b. click on either the `deploy.bat` (Windows) or 'deploy.sh' (Linux/MacOSX/wsl).



c. click on the download button and select the project folder as the location to download the file to. If the file is automatically downloaded to a downloads folder, move the file to the project folder.



3. within the project folder, create a folder called `inputs` and place all the appropriate data sets into this `inputs` folder

To run the app, navigate inside of the project folder and run (typically double click) on the deploy script. Upon doing so, you will be presented with a directory selection window that is prompting for the path of the project folder. Navigate to and select the project folder before clicking the "OK" button. Shortly thereafter, the application will appear in a browser tab.

> More specific information about the `deploy.bat` script
>
> The `deploy.bat` script performs the following:
>
> 1. defines paths to the project repository and local project folder
> 2. checks if `docker` is installed and available from the command line for the current user
> 3. checks if `docker` is running
> 4. query the user for the location of the project folder
> 5. determine whether there are any updates to the `docker` image and if so pull them down
> 6. run the `docker` container
> 7. open the shiny app in a browser

The Darwin Harbour Sediment Monitoring Program Analysis App

This Shiny application is designed to ingest very specifically structured excel spreadsheets containing Darwin Harbour sediment monitoring data and produce various analyses and visualisations. The application is served from a docker container to the localhost and the default web browser.

Docker containers can be thought of a computers running within other computers. More specifically, a container runs an instance of image built using a series of specific instructions that govern the entire software environment. As a result, containers run from the same image will operate (virtually) identically regardless of the host environment. Furthermore, since the build instructions can specify exact versions of all software components, containers provide a way of maximising the chances that an application will continue to run as designed into the future despite changes to operating environments and dependencies.

This shiny application comprises five pages (each accessable via the sidebar menu on the left side of the screen):

1. a **Landing** page (this page) providing access to the settings and overall initial instructions
2. a **Dashboard** providing information about the progression of tasks in the analysis pipeline
3. a **Data** page providing overviews of data in various stages
4. an **Exploratory Data Analysis** page providing graphical data summaries
5. an **Analysis** page providing information about the statistical models and their outputs
6. a **Manual** page that displays the online manual for the application

Each page will also contain instructions to help guide you through using or interpreting the information. In some cases, this will take the from of an info box (such as the current box). In other cases, it will take the form of little symbols whose content is revealed with a mouse hover.

There are numerous stages throughout the analysis pipeline that may require user review (for example examining the exploratory data analysis figures to confirm that the data are as expected). Consequently, it is necessary for the user to manually trigger each successive stage of the pipeline. The stages are:

- Stage 1 - Prepare environment

  More info

  This stage is run automatically on startup and essentially sets up the operating environment.

- Stage 2 - Obtain data

  More info

  This stage comprises of the following steps:

  - reading in the excel files within the nominated input path
  - validating the input data according to a set of validation rules
  - constructing various spatial objects for mapping and spatial aggregation purposes

  The tables within the **Raw data** tab of the **Data** page will also be populated.

- Stage 3 - Process data

  More info

  This stage comprises of the following steps:

  - apply limit of reporing values (LoRs)
  - pivot the data into a longer format that is more suitable for analysis and graphing
  - join in the metadata to each associated sheet
  - make a unique key
  - collate the all the data together from across the multiple sheets and files into a single data set
  - incorporate the spatial data
  - tidy the field names
  - apply data standardisations
  - create a site lookup table to facilitate fast incorporation of spatial information into any outputs.

  The tables within the **Processed data** tab of the **Data** page will also be populated.

- Stage 4 - Exploratory data analysis

  More info

  This stage comprises of the following steps:

  - retrieve the processed data.
  - construct spatio-temporal design plots conditioned on initial sampling semester
  - construct variable temporal design plots conditioned on harbour zone
  - construct site level temporal trends for each variable
  - construct zone level temporal and spatial visualisations for each variable

  The exploratory data figures of the **Exploratory Data Analysis** page will also be populated.

- Stage 5 - Temporal analyses

  More info

  This stage comprises of the following steps:

- retrieve the processed data
- prepare the data for modelling
- prepare appropriate model formulae for each zone, variable, standardisation type
- prepare appropriate model priors for each zone, variable, standardisation type
- prepare appropriate model template
- fit the models for each zone, variable, standardisation type
- perform model validations for each zone, variable, standardisation type
- estimate all the contrasts for each model and collate all the effects

Underneath the sidebar menu there are a series of buttons that control progression through the analysis pipeline stages. When a button is blue (and has a play icon), it indicates that the Stage is the next Stage to be run in the pipeline. Once a stage has run, the button will turn green. Grey buttons are disabled.

Clicking on button will run that stage. Once a stage has run, the button will change to either green (success), yellow (orange) or red (failures) indicating whether errors/warnings were encountered or not. If the stage was completed successfully, the button corresponding to the next available stage will be activated.

Sidebar menu items that are in orange font are active and clicking on an active menu item will reveal an associated page. Inactive menu items are in grey font. Menu items will only become active once the appropriate run stage has been met. The following table lists the events that activate a menu item.

Menu Item	Trigger Event
Landing	Always active
Dashboard	Always active
Data	After Stage 2
Exploratory Data Analysis	After Stage 4
Analysis	After Stage 5
Manual	Always active

Figure 2 provides a schematic overview the sequence of filesystem events that occur during the development, deployment and running of this app.

1. the developed codebase is pushed to github and if necessary continuous integration (github actions) is triggered. The continuous integration will re-build and host a docker image as well as rebuild the manual.
2. when the client runs the `deploy.bat` (or `deploy.sh`) script, it will check whether docker is running and get input from the user about the location of the project directory.
3. github will be queried to discover if a new docker image is available. If so, then the new image will be pulled down locally and run (if docker is runnning).
4. the docker container will be run and this will trigger git within the container to pull down the latest version of the codebase from github to a temporary repo in the container. As the container is starting up, it will mount the project folder so that its contents are available to the environment within container and outputs produced within the container are available to the host.
5. some of the files in the temporary repo will be copied to a folder within the project folder.
6. the shiny app will start up on `port 3838` of the localhost and this will be offered to the default browser.
7. as the shiny app progresses through each of the analysis stages, more data will be added to various folders of the project directory.

Analysis stages

Stage 2 - obtaining the data

At the completion of this stage, the Data sidebar menu and Stage 3 button will become active and the Data page will be populated with the raw data and available for review.

Read input info

This task seeks to determine what data sources are available and for those found, stores the names and filetypes discovered. This task will exclusively search in the `/input` folder of the project directory.

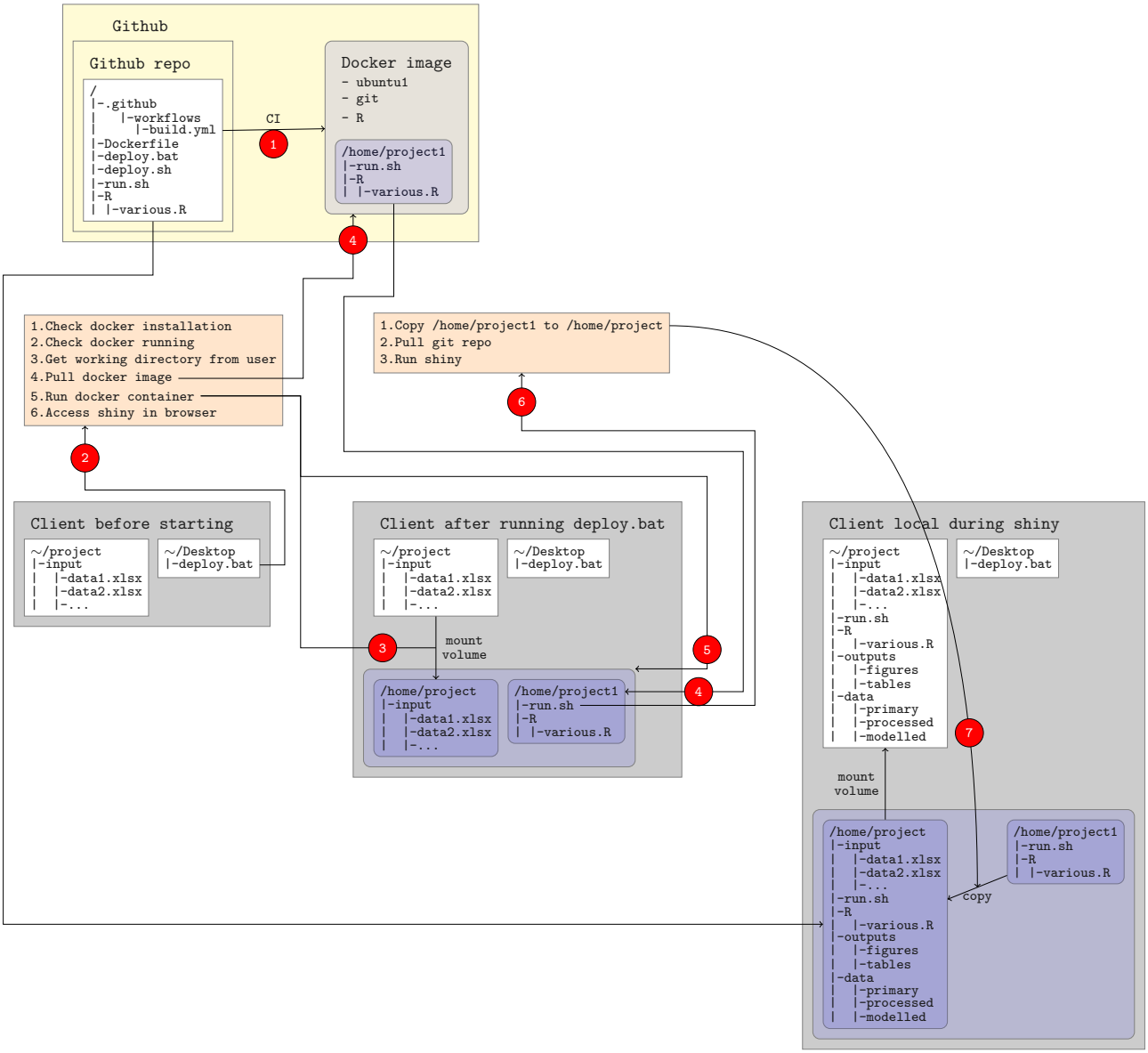


Figure 2: Diagram illustrating the sequence of filesystem events that occur during the development, deployment and running of this app.

Read input data

This task will sequentially read in each sheet of each data source (excel file) into a nested list.

Fix dates

This task will ensure that all dates are of the same format. Spreadsheets often store date/time data in one format and display it in another format. Consequently, users can be unaware that they have a mixture of date/time formats present in the same spreadsheet. For the purpose of data analysis, it is important that all date/time formats are consistent - this task aims to achieve this.

Validate input data

This task performs data validation in accordance with the rules set out in the following section.

Data requirements

To be valid, input data must be excel files (*.xlsx) comprising at least the following sheets (each of which must at least have the fields listed in their respective tables):

- metals

Field	Description	Validation conditions
Sample_ID	unique sample ID	must contain characters
*[1] (mg/kg)	observed concentration of metal in sediment sample	must contain only numbers or start with a '<' symbol

1: where the '*' represents a one or two character chemical symbol (such as 'Ag' or 'V'). There should be numerous of these fields

- hydrocarbons

Field	Description	Validation conditions
Sample_ID	unique sample ID	must contain characters
>C*[1]	observed concentration of hydrocarbons within a specific size bin in sediment sample	must contain only numbers or start with a '<' symbol

1: where the '*' represents a string of characters defining the size bin (such as '10 _C16'). There should be numerous of these fields

- total_carbons

Field	Description	Validation conditions
Sample_ID	unique sample ID	must contain characters
TOC (%)	observed total organic carbon (as a percentage of the sample weight)	must contain only numbers

- metadata

Field	Description	Validation conditions
IBSM_site	name of the site from the perspective of IBSM	must contain characters (or be blank)
Site_ID	a unique site ID	must contain characters (cannot be blank)

Field	Description	Validation conditions
Sample_ID	unique sample ID (the key to data sheets)	must contain characters (cannot be blank)
Original_SampleID	unique sample ID	must contain characters
Latitude	site latitude	must be numeric (and negative)
Longitude	site longitude	must be numeric
Acquire_date_time	date and time sample was collected (D/M/YYYY hh:mm:ss)	must be in datetime format
Sampler	name of person responsible for collecting sample (ignored)	ignored
Notes	project description (ignored)	ignored
Baseline_site	the unique site ID of the corresponding baseline sample	must contain characters (cannot be blank)
Baseline_acquire_date_site	the date and time of the corresponding baseline sample	must be in datetime format

- notes - this sheet is not processed or validated

> **Note**
>
> All input data must be placed in the `/input` folder of the Project prior to starting the app.

Make spatial data

This task will compile a set of spatial artifacts from GIS shapefiles of Darwin Harbour. These spatial artifacts will be used to spatially join the sediment data in order to assign spatial scales such as Zones and Areas to the data. They will also be used to facilitate mapping of the data. The shapefiles used in this task are built into the app. If there is a need to change these, please contact the app author.

Make spatial lookup

This stage creates a lookup table that relates each of the spatial scales to one another. This lookup is used to inject the spatial information into the data and modelled derivatives after they are created and in so doing prevents the need to spatially join the data each time it is required.

Stage 3 - processing the data

At the completion of this stage, the Stage 4 button will become active and the Processed Data sub-page of the Data page will be populated with the processed data and available for review.

Retrieve data

This task literally just reads in the data stored at the end of the previous stage.

Apply LoRs

This task applies rules for the presence of data that are below Limit of Reporting (LoR). In the data, LoR values are indicated by the presence of a < symbol. There are two ways available for handling LoR values.

1. Traditionally, values that represent Limit of Reporting (LoR) were replaced with half the LoR value. For example, a value of <0.02 would be replaced with 0.01.
2. However, modern statistical analyses have more appropriate ways of incorporating LoR information. Rather than arbitrarily replace values with half the LoR, we retain their value and flag them as censored and allow the statistical properties of disbutions to handle them more naturally.

In either case, a LoR flag is then attached to any value that was deemed LoR.

Pivot data

This task pivots (reshapes) the data from wide to long format. Wide format, in which each row represents a single site/time and each variable is in its own column is a convenient way to assemble data (particularly as it permits the user to easily identify missing values). However, data analysis requires that each individual record (observation) be in its own row.

Join metadata

This task joins (merges) the each of the main sediment data sheets (metals, hydrocarbons and total carbons) with the metadata sheet.

Make sample key

This task generates a unique key to uniquely identify each individual record by combining information about the Site_ID, acquire date/time and the part of the sample ID that indicates whether or not the sample was a replicate or duplicate.

Collate data

This task combines all the data sources (years) and types (metals, hydrocarbons, total carbons) together into a single data set. At the same time, it also creates some additional fields:

- `Year_cal` a field that represents the calendar year in which the sample was collected
- `Year_fiscal` a field that represents the fiscal year in which the sample was collected
- `Year_water` a field that represents the water year (defined as 1st Oct through to 30 Sept) in which the sample was collected
- `Baseline` a field that represents whether the observation is considered a "Baseline" observation or not
- `Replicate_flag` a field that represents whether the observation is a replicate
- `Duplicate_flag` a field that represents whether the observation is a duplicate

Incorporate spatial data

This task uses the spatial artifacts created in the previous stage to add spatial information to the data. This spatial information includes the Zone, Area and Site that each observation belongs to.

Tidy data

This task creates a new field `Site` that acts as a unique identifier of a sampling location over time. This field is created by copying the `IBSM_site` field (if it is not empty), otherwise the `Site_ID` field is used. This task also removes any fields that are no longer required.

Standardise data

All sediment metals and hydrocarbons are expressed and analysed in both raw and standardised forms. Data standardisation rules are summarised in Figure 3.

In the case of Fe/Al standardisations (which apply to Ag, Co, Cu, Hg, Ni, Pb and Zn):

1. Fe/Al ratios are calculated per sample (observation).
2. A five year rolling mean is then applied to the ratios to dampen extreme fluctuations
3. Using a threshold of Fe/Al < 1.3, each sample is tagged according to which of either Fe or Al it would originally be standardised against (if Fe/Al < 1.3, Al is used otherwise Fe).
4. Within, each site, all values should be standardised the same way (e.g. either against Fe or Al, but not a mixture). Hence, the most common form of standardisation is determined and used to standardise all observations in the series for a given site.
5. Standardisations against Al are $val \times (50,000/Al)$

6. Standardisations against Fe are $val \times (20,000/Fe)$
7. During the step above, if the standardisation of an individual observation was changed, then it is flagged

Data standardisation rules for As:

1. Inner Harbour samples are standardised against Fe ($val \times (20,000/Fe)$)
2. Outer Harbour samples are standardised against Mn (val/Mn)
3. An alternate form of standardation is also performed in which all As samples are standardised against Fe ($val \times (20,000/Fe)$)

Vanadium is always standardised against Fe ($val \times (20,000/Fe)$).

Finally, all hydrocarbons are standardised against Total Organic Carbon (TOC) (val/TOC).

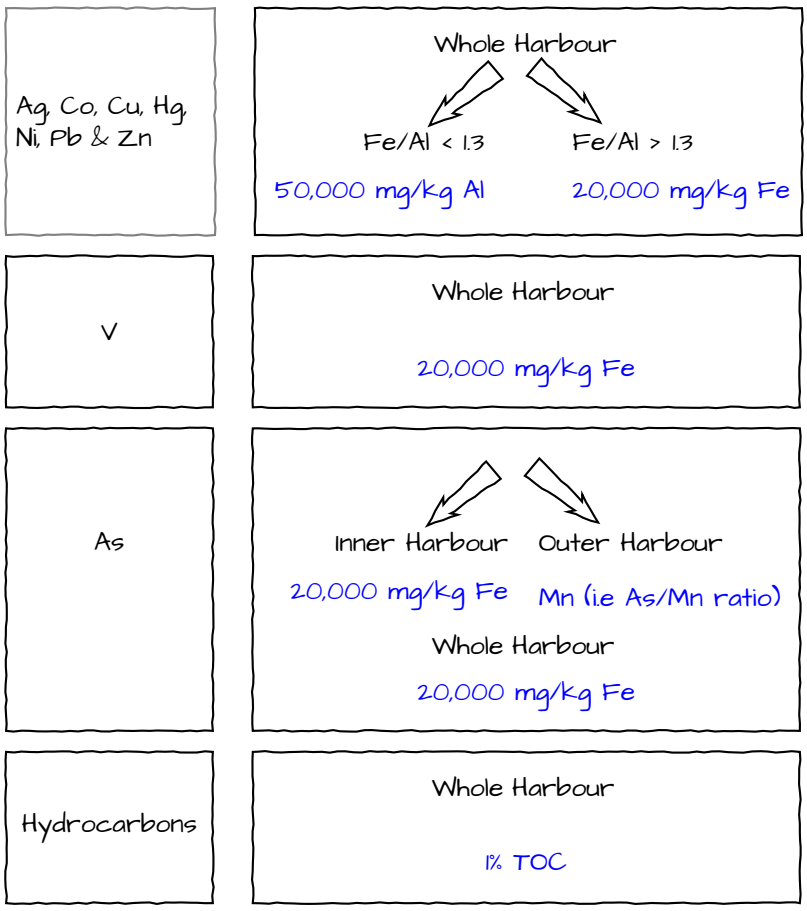


Figure 3: Diagram illustrating the standardisation (normalisation) rules applied to each variable. In each case, the text in blue represents the appropriate divisor used in the standardisation.

Create site lookup

This task creates a lookup that maps sites to zones.

Stage 4 - Exploratory data analysis

At the completion of this stage, the Exploratory Data Analysis menu and Stage 5 button will become active and the Exploratory Data Analysis page will be populated with the a range of exploratory figures. This Stage involves numerous tasks that each prepare the data in formats conducive to the production of the figures while navigating the Exploratory Data Analysis page.

Stage 5 - Temporal analysis

At the completion of this stage, the Analysis menu will become active and the Analysis page will be populated with the a range of modelled outputs.

The temporal analyses essentially involve the fitting of separate Bayesian Hierarchical models (Gelman and Hill 2007) to the full time series of all sites within each focal Zone. From such models (outline below), site and zone level modelled trends can be inferred and thereafter aggregated up to Area and Whole of Harbour level trends as well.

i Bayesian hierarchical models

Hierarchical models are used to analyse data that are structured such that there are sampling units nested within higher order units in a hierarchical manner. For example, if there are multiple sampling stations within each sampling site and there are multiple sites within each region, then we can describe the sampling design as being nested or hierarchical. Such designs usually provide greater statistical power (ability to detect effects when they occur) for a given sampling effort by reducing sources of unexplained variability. However, such designs must also be analysed carefully to avoid psuedo-replication (e.g. when subsamples are elevated to the status of full samples, thereby erroneously increasing statistical power) and confounding. For example, observations collected from multiple stations are not independent of one another at higher spatial scales (that is, the observations collected at the stations within one site are going to be more similar to one another than they are to the observations collected at stations within another site. Such dependencies structures must be carefully considered in order to yield reliable model outcomes. Similarly, repeated sampling within a location will also yield non-independent observations.
Unlike traditional *frequentist* methods, in which probability is calculated as the expected long-run chance of obtaining the observed data and thereby pertaining only to the extremity of the data, in *Bayesian* methods, probability pertains directly to the underlying parameters and hypotheses of interest. Rather than generating point estimates or p-values (which are themselves commonly missused and missunderstood), Bayesian methods offer more intuitive, probabilistic interpretations of results, such as the probability that certain effects or events occur. Bayesian methods can also incorporate prior beliefs about the parameters, which can be particularly useful when data are sparse or noisy.
Hence, Bayesian statistics offers a flexible and comprehensive toolkit for making inferences and predictions, handling uncertainty, and incorporating expert knowledge into the analysis of data and provide intuitive interpretations that relate directly to the underlying scientific or management questions and that are accessable to a broad audience.
In essence, frequentist statistics calculates the probability of the data given a hypothesis ($P(D|H)$) - this is why frequentist conclusions pertain to the data and not directly to the hypotheses. Arguably, it would be more useful to express statistical outcomes as the probability of hypotheses given the available data ($P(H|D)$). **Bayes' Rule** is a fundamental principle in probability theory that describes the conditional relationship between these two and outlines how the later can be obtained from the former. It calculates the probability of a hypothesis given observed evidence ($P(H|D)$ - the *posterior probability* of the hypothesis) by combining the likelihood of the observed evidence five the hypothesis ($P(D|H)$), the *prior* probability (belief) of the hypothesis (before seeing the evidence, $P(H)$), and the overall probability of observing the evidence under any hypothesis ($P(D)$) according to the following:

$$P(H|D) = \frac{P(D|H).P(H)}{P(D)}$$

The seeming simplicity of the above conditional probability equation belies the underlying intractability for all but the most trivial of use cases. In order for $P(H|D)$ to be a probability distribution (and thus useful for drawing conclusions), it is necessary that the area under the distribution must add up to exactly 1. This is the job of the divisor in the equation above - to act as a normalising constant. Unfortunately, in most cases, it is not possible to calculate this normalising constant. It is for this reason that Bayesian statistics (which actually predated frequentist statistics) remained dormant for more than a century.
The advent of modern computing along with the cleaver application of a couple of mathematical techniques has since revised Bayesian statistics. Rather than attempt to calculate the normalising constant, modern Bayesian techniques aim to reconstruct the unknown posterior distribution ($P(H|D)$) by repeatedly sampling from the product of $P(D|H)$ and $P(H)$. This technique, Markov Chain Monte Carlo (MCMC) provides a powerful and flexible way to estimate the posterior distributions of model parameters, especially when these distributions are complex and cannot be solved analytically.
Following is a high-level overview of the general MCMC sampling:

1. The process begins with an initial estimate of the parameters based on prior knowledge or assumptions.
2. MCMC then generates a series of estimates, or "draws," for each parameter. Each new estimate is made based on the previous one, creating a chain. The way each new estimate is made ensures that, over time, more probable estimates are chosen more often than less probable ones.
3. With each estimate, the model evaluates how well it explains the observed data, using the likelihood of the data given the parameters. It also considers the prior belief about the parameters. This combination of data fit and prior belief guides the process toward more probable parameter values.
4. Initially, the estimates might be far off, but as the chain progresses, it starts to settle into a pattern. This pattern represents the posterior distribution of the parameters - a probability distribution that reflects both the observed data and the prior information.
5. After a pre-defined number of steps (assuming the chain has reached equilibrium), the draws are considered to be representative of the posterior distribution. These draws allow us to estimate the parameters, their uncertainties, and any other quantities of interest.
6. Since the purpose of MCMC sampling is to reconstruct an otherwise unknown posterior distribution, it's crucial to check that the chain has fully explored and reconstructed the entire distribution. Typically, the MCMC process involves running multiple chains with different starting points and ensuring they all converge to the same distribution.

The general form of the models employed is as follows:

$$
\begin{aligned}
y_{i,s} &\sim \Gamma(\mu_{i,s}, \phi) \\
log(\mu_{i,s}) &= (\beta_0 + \gamma_{s[i],0}) + \sum_{j=1}^{n} T_{[i],j}.(\beta_j + \gamma_{s[i],j]}) \\
\phi &\sim \Gamma(0.01, 0.01) \\
\beta_0 &\sim t(3, \alpha_1, \alpha_2) \\
\beta_{[1,n]} &\sim t(3, 0, \alpha_3) \\
\gamma_{[1..p]} &\sim MVN(0, \Sigma_s) \\
\Sigma_s &= \begin{pmatrix} \sigma^2_{s_1} & \rho_s\sigma_{s_1}\sigma_{s_2} & \rho_s\sigma_{s_1}\sigma_{s_3} \\ \rho_s\sigma_{s_1}\sigma_{s_2} & \sigma^2_{s_2} & \rho_s\sigma_{s_2}\sigma_{s_3} \\ \rho_s\sigma_{s_1}\sigma_{s_3} & \rho_s\sigma_{s_2}\sigma_{s_3} & \sigma^2_{s_3} \end{pmatrix} \\
\sigma_{s[1,2,3]} &\sim t(3, 0, \alpha_2) \\
\rho_s &\sim LKJcorr(1)
\end{aligned}
$$

The i_{th} value (y) from the s_{th} site was assumed to be drawn from a gamma (Γ) distribution parameterised by a mean ($\mu_{i,s}$) and dispersion (ϕ) respectively. The (natural log) expected means were described by a linear model that included an intercept (β_0), varying effects of site ($\gamma_{s,0}$) and annual changes in value ($\gamma_{s,j}$) as well as the population effects (β_1) of year (T). Weakly informative flat-t (3 df) priors were applied to the intercept and all population effect parameters. The values (α_1, $alpha_2$ and α_3) used to define the weakly informative priors were developed from simple summary statistics of the raw data. A weakly informative gamma prior was applied to the dispersion parameter. The varying effects were assumed to follow a multivariate normal with a site-specific covariance structure whose variances follow a weakly informative flat t distribution and whose correlation follows a LJK distribution with parameter of 1.

When the data include values that are below the limit of detection, the model outlined above is modified so as to apply left censoring.

All Bayesian models were fit using the `brms` (Bürkner 2017) package within the R Graphical and Statistical Environment (R Core Team 2024). All models had an adaptive delta of 0.95 and had three chains, each with 5000 no-u-turn (NUTS) iterations, a warmup of 1000 and a thinning rate of 5.

Separate models are fit to each variable, for each appropriate standardisation type, for each zone. At the time of writing this manual, this equates to nearly 300 models.

Retrieve data

This task literally just reads in the data stored at the end of the previous stage.

Prepare data

This task ensures that the data are formatted and packaged up into sets associated with each individual model.

Prepare priors

This task is responsible for defining weakly informative priors on all parameters for each model. The priors associated with the model for a specific variable/zone were developed by taking simple summary statistics of the mean, median, standard deviation and median absolute deviation of the log of the values conditional on year.

- α_1 was taken from the median of the log values from the first sampling year. This was used as the mean of the model intercept (β_0) prior
- α_2 was taken from the maximum of the median absolute deviations of log values for each sampling year. This was used as the standard deviation for the model intercept (β_0) as well as the standard deviation for the variances (σ_s) of the varying effects.
- α_3 was taken as the maximum of the difference in mean log values between years and this was used to inform the standard deviation of the population effect (β) priors.

Prepare model template

This task essentially involves compiling a single simple model to use as a template for most other models. Model compilation consumes approximately 40 seconds of time prior to the model running. Since most of the models are the same (only the priors and the data differ), and this project requires the fitting of a very large number of models, the use of a pre-compiled template can speed up the overall modell fitting process dramatically.

Fit models

This task involves fitting all combinations of the models. As each new model is fit, the **Model Logs* pane of the** Dashboard** page will be updated with a running progress (model number out of a total), zone/variable/standardisation name along with a message indicating whether the model was run or retrieved from a previous run. Each single model is expected to take approximately 1 minute to run (depending on the clock speed of the computer) so adjust your expectations accordingly.

Validate models

This task will perform a range of model validation checks and assign flags against models that display sub-optimal characteristics. Similar to the model fitting task, the status of validation can be tracked in the **Model Log** pane of the **Dashboard** page. Details of the validations performed are given in section Section .

Compile zone results

This task will extract posteriors and summaries for model derived cell means (estimates for each year) for each zone along with effects (comparisons between sets of years). With respect to the effects, the comparisons are:

- Baseline to each subsequent year
- Most recent year to the year prior to that

The full posteriors of each of the above are stored in files to be accessed from the **Analysis** page.

Collect zone results

This task collects file pointers across all models together into a single file for more convenient access in the **Data** page.

Compile site results

Similar to the **Compile zone results** this task extracts posteriors and summaries for model derived cell means (estimates for each year) for each site along with effects (comparisons between sets of years). With respect to the effects, the comparisons are:

- Baseline to each subsequent year
- Most recent year to the year prior to that

The full posteriors of each of the above are stored in files to be accessed from the **Analysis** page.

Collect site results

This task collects file pointers across all models together into a single file for more convenient access in the **Data** page.

Compile area results

This task aggregates the zone level posteriors together before re-calculating the cell means and effects.

Collect area results

This task collects file pointers across all models together into a single file for more convenient access in the **Data** page.

Finally all site, zone and area results are concatenated together into a single file.

App pages

Landing page

To run this tool, please adhere to the following steps:

1. review the *Path Settings* (specifically checking the **"Data input dir"** and ensuring that there is at least one data file listed in the box under this setting
2. review the *Run Settings*. In particular,
    - consider whether you need to **Clear the previous data** - clicking the button to do so. Clearing the previous data deletes all cache and ensure that the analyses are performed fresh. **This is recommended whenever the input data changes**. Not clearing the previous data allows the user to skip directly to later run stages if earlier stages have already been run.
    - consider the Limit of Reporting (LoR) setting.
        - the default is to set the value equal to the specified limit of reporting for a value (such values must start with a "<") and will be flagged as "left" censored. Models that accommodate censored data take a probabilistic approach to inferring the likely distribution of all observations including those beyond the limit of reporting and are considered more appropriate.
        - the alternative is the more traditional approach of replacing the value with 1/2 of the limit of reporting value and using this in the analyses. Whilst traditional, this approach tends to make the resulting values into outliers and thus problematic in analyses.
3. navigate the *Dashboard* via the menu on the left sidebar

Dashboard

The analysis pipeline comprises numerous **Stages**, each of which is made up of several more specific **Tasks**. The individual Tasks represent an action performed in furtherance of the analysis and of which there are reportable diagnostics. For example, once the application loads, the first Stage of the pipeline is to prepare the environment. The first Task in this Stage is to load the necessary R packages used by the codebase. Whilst technically, this action consists of numerous R calls (one for each package that needs to be loaded), the block of actions are evaluated as a set.

Initially, all upcoming tasks are reported as "pending" (). As the pipeline progresses, each Task is evaluated and a status is returned as either "success" () or "failure" ().

The Stage that is currently (or most recently) being run will be expanded, whereas all other Stages will be collapsed (unless they contain errors). It is also possible to expand/collapse a Stage by double clicking on its title (or the small arrow symbol at the left side of the tree).

As the pipeline progresses, Task logs are written to a log file and echoed to the **Logs** panel. Each row represents the returned status of a specific Task and are formatted as:

- the time/date that the Task was evaluated
- the Task status, which can be one of:
  - `SUCCESS` the task succeeded
  - `FAILURE` the task failed and should be investigated
  - `WARNING` the task contained a warning - typically these can be ignored as they are usually passed on from underlying routines and are more targetted to developers than users.
- the Stage followed by the Task name
- in the case of errors and warnings, there will also be the error or warning message passed on from the underlying routines. These can be useful for helping to diagnose the source and cause of issues.

The Logs in the Log panel are presented in chronological order and will autoscroll such that the most recent log is at the bottom of the display. If the number of Log lines exceeds 10, a scroll bar will appear on the right side of the panel to help reviewing earlier Logs.

> Note
> The Status and Logs are completely refreshed each time the application is restarted.

The Progress panel also has a tab (called **Terminal-like**) which provides an alternative representation of the status and progress of the pipeline.

Under the **Logs** panel, there is a **Model Logs** panel. This panel provides additional status and progress about the fitting and processing of individual statistical models.

Data

The Data page comprises two panels or subpages accessable by tabs named "Raw data" and "Processed data" at the top of the page.

> **Note**
>
> The contents of the Processed data subpage will not be revealed until the completion of Stage 3.

Raw data panel

The **Raw data** panel displays the input data and associated validation summaries (once the data have been loaded - that is, once Stage 2 has been complete). The table above initially has a row for each of the input files.

The title of each input file name is displayed in the first column (**File**). The size and file creation time in the next two columns (fields). The **Sheet** field lists the parsed sheets within the excel file and the **Status** column indicates whether all the sheets are valid () or not ().

To the left of the file name there is a black triangle. This is an content expansion marker. When the triangle points to the right, clicking anywhere in the cell containing the triangle will expand the table to reveal additional rows (one for each of the sheets in that excel file). The rows can be collapsed again by clicking on the cell containing the downward pointing triangle.

When the additional rows are visible, the **Status** field icons indicate whether the sheet was valid () or not ().

Clicking on the cell containing the name for a Sheet will make this the *focal* Sheet of the **Data** and **Validation** tabs:

- the table in the **Data** tab displays the *focal* content of the input data Sheet. Only the first 10 rows are displayed in the table, the others being accessable via the controls under the table.

  Note, all numerical values are displayed only to three decimal places, yet the actual underlying data is full resolution.

- the table in the **Validation** tab displays more details about which fields or rows of the *focal* Sheet failed validation tests.

  If there were no validation issues, this table will be empty. Otherwise, the description field will indicate the nature of the violation and in the case of issues with an individual record, the offending row will be presented across the remaining cells in the row. For more information about the validation tests, please refer to the **Data requirements** box (to the right of this box in the app).

Underneath both the Data and Validation tables, there is a **Download as csv** button. Via this button, you can download a comma separated text file version of the data in the table for further review in a spreadsheet of your choice. Once you click this button, you will be prompted to navigate to a suitable location to store the file.

Processed data panel

The Processed data panel displays the first 10 rows of the complete, compiled and processed data set. Descriptions of each field of these data are provided in the table below.

> **Note**
>
> This panel will not become active until the completion of Stage 3.

The **Processed data** panel displays the processed data. As part of the processing, the following new fields will be created:

Field	Description
Sample_key	this is a unique ID for each sample and is created as the combination of `Sample_ID`, `Acquired_date_time` and any replicate/duplicate tokens in the `Sample_ID`
*[1] (mg/kg)	observed concentration of metal in sediment sample
Type	whether the record pertains to a metal or hydrocarbon
Year_cal	calendar year of sample collection
Year_fiscal	financial year of sample collection
Year_water	water year (1st Oct - 30 Sept) of sample collection
Year	calendar year of sample collection
Baseline	whether the record is a baseline record (TRUE) or not (FALSE)
ZoneName	name of the Darwin Harbour Zone
Region	Darwin Harbour Region number
RegionName	name of the Darwin Harbour Region
Zone	Darwin Harbour Zone number
Area	Darwin Harbour Area (Inner or Outer)
Site	ID of the sampling site
Var	name of the measurement
Values	the observed measurement
Value_type	whether the value is a standardised value or not
Fe/Al	Fe:Al where appropriate

Field	Description
Fe_Al_normalisation	what the sample **would be** normalised against Fe or Al (or not applicable) if it could be normalised
Normalised against	what the sample **was** normalised against (if it was normalised)
Normalisation flag	whether the normalisation has switched between Fe and Al over time for this site

Under the column (field) headings in the Processed data panel table, there are input boxes that act as filters. The data presented in the table will be refined to just those cases that match the input string as it is being typed. It is possible to engage with any or all of these filters to help refine the search.

Under the table there is a **Download as csv** button. Via this button, you can download a comma separated text file version of the data in the table for further review in a spreadsheet of your choice. Once you click this button, you will be prompted to navigate to a suitable location to store the file.

Exploratory Data Analysis

The Exploratory Data Analysis page comprises four panels or subpages accessable by tabs at the top of the page and named "Temporal", "Temporal Type", "Temporal Site" and "Spatial Type".

Temporal

This page displays a multi-panel figure depicting the temporal sampling design of the data within each of the Harbour Zones. The y-axis (rows) of the figure represents the Sampling sites (based on the Site names they were first assigned). Blue points represent the samples collected that are considered to be "Baseline" or "Reference" samples from which subsequent samples at the corresponding site are to be gauged (compared). Red points represent non-"Baseline" samples. Points are jointed by lines to help identify discontinued sampling (where no line exists) and where no Baselines have been defined (when the left point of a sequence is red).

To keep the figure to a manageable size and degree of clutter, the figure fucusses only on "Sites" that were first monitored in the semester indicated by the dark blue selector to the left side of the figure. The selectors are partitioned into 6 monthly (a semester) chunks and clicking on a different selector will update the figure to a different fraction of the data.

Temporal Type

The multi-panel figure displayed on this page partitions the data into Zone/Measurement types (via left hand selectors) and figure columns (measures). The x-axis (rows) of the figure represents the sampling years and the y-axis represents the individual sampling sites. Blue points represent the samples collected that are considered to be "Baseline" or "Reference" samples from which subsequent samples at the corresponding site are to be gauged (compared). Red points represent non-"Baseline" samples.

Temporal Site

This page provides a way of fully exploring the un-modelled temporal trends in both raw and standardised data for each Site, Measurement Type and Variable.

Blue points represent the samples collected that are considered "Baseline" or "Reference" samples from which subsequent samples at the corresponding site are gauged. Red points represent non-"Baseline" samples. Multiple panels (if present) distinguish standardised and unstandardised data. Hovering over the points will reveal some of the pertinent underlying data associated with the point.

Different combinations of Sites, Measurement Type and Variables can be selected via their respective dropdown boxed in the panel above the figures.

Spatial Type

This page provides a way to visualise the spatial distribution of un-modelled data on a map. It also provides a representation of the temporal trends in the data. The subset of data presented in the temporal graph and on the map are controlled by a set of dropdown selector boxes. The data presented on the map are also further refined to a single year and that is controlled by the slider situated above the temporal figure.

In the temporal figure, each point represents an individual observation and observations collected from the same sites over time are connected by gray lines. Blue points represent the samples collected that are considered "Baseline" or "Reference" samples from which subsequent samples at the corresponding site are gauged. Red points represent non-"Baseline" samples. Multiple panels (if present) distinguish standardised and unstandardised data. Hovering over the points will reveal some of the pertinent underlying data associated with the point.

Sites are represented on the map by circular points, the colour of which are mapped to a scale proportional to the data values. Panning and zooming of the map is done via the mouse (moving while left mouse press: panning, scrolling of mouse button: zooming). The Zone selector dropdown controls which Harbour Zone(s) are highlighted and populated with data.

Analysis

The Analysis page comprises three panels or subpages accessable by tabs at the top of the page and named "Analysis overview", "Model diagnostics" and "Analysis details".

Analysis overview

The main feature of this panel is a table representing a very high level overview of the results conditional on spatial scale (Site, Zone, Area), Measurement Type (hydrocarbons or metals), Value Type (Unstandardised or Standardised), Normalisation (Standardisation) method, Focal Year and Contrast (each of which is controlled via a dropdown).

The table presents a matrix in which:

- the columns represent the variables
- the rows represent the sites/zones/areas
- the colour of the cells represents the polarity and evidence of change:
  - red: strong evidence of an increase
  - orange: evidence of an increase
  - yellow: weak evidence of an increase
  - white: no evidence of change
  - light green: weak evidence of a decline
  - mid green: evidence of a decline
  - dark green: strong evidence of a decline
  - gray: model not completed (typically due to a lack of data)
- models built on data that includes values below limit of reporting/detection are marked with a black flag ()

The selectors can be used to filter what subset of data are presented in the table:

- scale of aggregation (site/zone/area)
- whether or not the data are standardised
- the type of variable (metals, hydrocarbons or both)
- the focal year to compare to the baseline (2019/2020)

At the top of this page there is a collapsed box containing an overview of the model validation checks. To view the table, ensure that the box is expanded (click on the "-" icon in the upper right corner of the box banner).

The table is filterable (by entering text in the text boxes between the column headings and first row of data). Each row represents the inforation about a single model. The validity tests represent whether there is any evidence to reject (p-value < 0.05). A value of "fail" indicates that there is evidence that the test revealed evidence to invalidate the assumption:

- **ks** the KS (uniformity test)
- **ds.p** the dispersion test
- **q** residual plot quantiles
- **o** outlier observations
- **valid** whether any of the above failed

The table is initially sorted such models flagged as potentially invalid are towards the top.

The page also displays diagnostics and summaries of each of the fitted models accessible via hierarchical tabs and selectable via Zone, Variable and Standardisation selection boxes.

More information about each of the validation and summarisations are provided inside the corresponding set of tabs.

Downloading modelled outputs

All the summarised modelled effects can be downloaded as a single comma separated (csv) file via the blue **Download effects as csv** button. Upon clicking the button, a dialog box will appear from which you can select a filename (a name that includes the current date is provided by default) and path. Since the Longitude and Latitude for numerous sites appears to have drifted over time, the Longitude and Latitudes included in the downloaded file represents the mean Latitude and Longitude for the Site. At this stage, Longitude and Latitude are not provided for Zone and Area scale model estimates.

Similarly, all of the summarised modelled cell means (trends) can be downloaded as a single comma separated (csv) file via the **Download trends as csv** button. Notes on Longitude and Latitude are as above.

> **!** Important
>
> The routine behind this later button involves compiling together all posteriors across all scales and summarising each of them. These steps take a few minutes to complete. During this time, you are free to navigate around the application. However, be mindful that at some time a dialog box will eventually popup and assume focus. Hence, if you are not paying attention, you may accidentally change the filename and/or path, or else cancel the download event altogether - in which case, you will need to ask it to download again and it will need to re-compile and summarise the modelled results all over.

Model diagnostics

This panel displays a wide range of MCMC sampling and model validation diagnostics as well as simple model summaries.

Model Validation

Prior vs Posterior

Plots of priors (black points and whiskers) and posteriors (coloured points and whiskers) for each of the main model parameters. This sort of figure provides a way to visualy assess how **informative** the priors are likely to be in determining the posteriors. Ideally, we want the priors to only be **weakly** informative. That is, we only want them to have a regularising influence (encouraging the sampler to stay within a range of parameter estimates that are vaguely plausible). Hence, the ideal is for the priors (black) to be either substantially wider (longer in the figure) or centered at a different location (position along y-axis) than the corresponding posteriors (coloured).

Traceplots

MCMC traceplots are visual tools used to assess the convergence and mixing behavior of Markov Chain Monte Carlo (MCMC) algorithms in Bayesian statistics. They offer valuable insights into the efficiency and reliability of your MCMC analysis.

What do they show?

Each trace plot displays the values of a specific parameter estimated by the MCMC algorithm across all iterations. Multiple parameters can be plotted individually or jointly to compare their behavior.

How to interpret them:

- **Convergence:**

  A well-converged chain should exhibit stability over iterations, meaning the trace plot plateaus around a constant value or fluctuates within a predictable range. Trends or drifts indicate the chain hasn't converged, and the estimates might be biased.

- **Mixing:**

  Good mixing implies the chain explores the parameter space efficiently, visiting different possible values frequently. Stuck chains remain in specific regions, leading to poor exploration and unreliable estimates.

- **Autocorrelation:**

  If consecutive values in the trace plot are highly correlated, the chain mixes slowly, impacting efficiency.

- **Mixing across chains:**

  Running multiple chains (starting from different points) should converge to similar values, supporting mixing and reliability.

Interpretation tips:

- Consider the expected behavior of your model: some parameters might naturally fluctuate more than others.
- Use reference values or theoretical limits to judge if the parameter values seem reasonable.
- Combine trace plots with other diagnostics like Gelman-Rubin convergence measures for a comprehensive assessment.

Autocorrelation plots

MCMC autocorrelation plots, are another essential tool for diagnosing the convergence and mixing of your MCMC algorithm in Bayesian statistics.

What are they?

Autocorrelation plots measure the correlation between values of a parameter at different lags (distances) within the MCMC chain. They typically display the correlation coefficient at each lag, plotted against the lag number. The first bar represents the correlation of MCMC samples with themselves, and thus will always be 1 on the y-axis.

How to interpret them:

- **Bias:**

  The autocorrelation plot helps assess how quickly the correlation between samples decays as the lag increases. Ideally, samples collected from any process (including MCMC sampling) should all be independent and unbiased in order to provide unbiased estimates. If the autocorrelation values decay rapidly to zero as the lag increases, it indicates that the MCMC samples are likely to be independent and unbiased. Conversely, if high autocorrelation persists with large lags, it is possible that the MCMC samples are biased towards particular regions of the parameter space. **Ideally, the degree of autocorrelation should drop to below 0.25 by the second bar (second lag).

- **Convergence:**

  - Rapidly decaying autocorrelation: As the lag increases, the correlation between values drops quickly, indicating good convergence. The chain efficiently moves through the parameter space, providing reliable and unbiased estimates.
  - High autocorrelation even at large lags: This suggests the chain is "stuck" in certain regions, not exploring the parameter space well. Estimates might be biased and unreliable.

- **Mixing:**
  - High autocorrelation at small lags: The chain takes many iterations to "forget" past values, indicating slow mixing. The algorithm might not be efficiently exploring the parameter space.
  - Autocorrelation dropping to zero around medium lags: This suggests the chain mixes reasonably well, considering the natural dependence between consecutive values.

If autocorrelation patterns exist in the MCMC samples, the MCMC samples should be thinned to a higher degree.

Rhat plots

Rhat plots, also known as Gelman-Rubin convergence diagnostic plots, are graphical tools used to assess the convergence of multiple chains in a Bayesian analysis. These plots are based on the Gelman-Rubin statistic (Rhat), which compares the variance within chains to the variance between chains.

What are they?

Rhat represents a "potential scale reduction factor", and compares the within-chain and between-chain variances of parameter estimates from multiple MCMC chains. It's used to assess whether the chains have mixed well and converged to the same target distribution.

Rhat plots visualise the Rhat statistic for each parameter estimated in the Bayesian analysis. Each parameter has its own Rhat value, and the plot typically displayed as histograms.

How to interpret them:

- **Rhat < 1.01:**

  Generally indicates very good convergence, suggesting the chains have explored the target distribution efficiently and reached similar conclusions.

- **Rhat < 1.05:**

  Generally indicates acceptable convergence.

- **Rhat > 1.05:**

  Indicates significant convergence problems, meaning the chains haven't mixed well and could lead to biased estimates. It implies that at least one of the chains may have traversed different features of the parameter space compared to the other chain(s). If so, then there may well be other additional un-traversed features. It is possible that the MCMC sampler may not have fully explored all important regions of the parameter space.

Effective Sample Size

ESS plots, or Effective Sample Size plots, are graphical tools used to assess the effective sample size of MCMC samples obtained from Bayesian analyses. The effective sample size quantifies the amount of independent information contained in the MCMC samples and is crucial for accurate estimation of posterior quantities.

How do they work?

Imagine your MCMC chain has length n, but due to autocorrelation, not all n iterations provide truly independent information. ESS attempts to quantify this by estimating the number of independent samples equivalent to your chain, effectively capturing the information content.

How to interpret them:

- **High ESS (closer to n):**

  This indicates good efficiency, meaning your chain efficiently explores the parameter space and provides reliable estimates.

- **Low ESS (much smaller than n):**

  Suggests poor efficiency, with high autocorrelation reducing the effective number of independent samples. This can lead to wider credible intervals and less precise estimates.

In order to have enough MCMC samples to provide meaningful summaries, each parameter should have at least 1000 effective samples.

The ESS values are often expressed as fractions of the total n and represented graphically as a histogram. Ideally, most (if not all) the ESS values should be above 0.5.

If there are low ESS, it suggests that either the total number of iterations was not large enough or the sampler was not sampling efficiently. For the former case, the sampler should be re-run with additional iterations. In the later case, it is likely that the model itself (or the priors) are mis-specified and this should be addressed before rerunning the model.

Posterior Probability plots

Posterior probability density overlay plots display the density distribution of the observed data (dark line) overlayed upon the density distributions of a large number of posterior predictions (fainter blue lines).

The general idea is that if a model is to be useful, it should be able to generate observations (predictions) that approximate the data used to train the model in the first place. If this is not the case, then it is likely that the model does not adequately capture the pertinent properties of the observed data.

Ideally, the densities of the numerous realisations from the posteriors should match closely the density of the original observed data. The manner by which the densities differ from that of the original data can be used to infer what aspects of the data the model is under or over estimating and adjustments can be made to the model accordingly.

Simulated (DHARMa) residuals

Since statistical models are low dimensional representations of a system the reliability of a statistical model will depend on the degree to which certain assumptions are met. Many of these assumptions can be explored by examination of the model residuals.

Patterns in residuals suggest either issues of dependencies (biases), poor model structure and a general lack of model fit. However, for many statistical models, discerning genuinely violation-indicating patterns in residuals from artifacts due to the model type can be really difficult - if not impossible.

Within R, the *DHARMa* (Diagnostics for HierArchical Regression Models: Hartig (2022)) package generates standardised residuals via simulation and uses these as the basis of a range of tools to diagnose common modelling issues including outliers, heterogeneity, over-dispersion, autocorrelation.

New observations simulated from the fitted model are used to calculate a cumulative density function (CDF) that describes the probability profile of each observation. Thereafter, the residual of an observation is calculated as the value of the CDF that corresponds to the actual observed value:

- a value of 0 indicates that the observed value was less than all simulated values
- a value of 1 indicates that the observed value was greater than all simulated values
- a value of 0.5 indicates that the probability of obtaining the observed value is 50%.

This approach ensures that all residuals have the same interpretation irrespective of the model and distribution selected.

DHARMa supports a variety of diagnostic plots based on the simulated residuals, including residual plots, QQ plots, and test-based diagnostic plots. These plots allow for visual inspection of model adequacy and can reveal patterns or deviations indicative of model misspecification or violation of assumptions. The most common plots and their interpretations are:

- a **Q-Q plot**

  Ideally all points should be close to the diagonal red line. Overlayed onto this plot are three additional tests.

1. KS (Kolmogorov-Smirnov) test investigates whether the (in this case simulated) are likely to have been drawn from the nominated distribution.
2. Dispersion test investigates whether the is any evidence of overdispersion (more variability than the model expects) estimated as the standard deviation of the data is equal to that of the simulated data)
3. Outlier test investigates for the prevalence of outliers (when observed values are outside the simulated range)

- a **Residual vs Predicted plot**

  Ideally, there should be no patterns in the residuals. To help identify any patterns, quantile trends are overlayed. Ideally, there should be a flat black line at each of the quantiles of 0.25, 0.5 and 0.75. In some circumstances, quantiles cannot be computed and in such cases a single dashed smoother many be placed over the data cloud.

- a **Dispersion plot**

  The observed model dispersion is overlayed (red line) upon the distribution (black) of simulated dispersion values. Ideally the red line should be in the middle of the simulated distribution.

  - If the red line is to the far right, the model is considered overdispersed. Parameter uncertainty is typically underestimated in overdispersed models - this leads us to be more confident in our results than we should be and this is bad.
  - If the red line is to the far left, the model is considered underdispersed. This is usually an artifact and results in our estimates being more conservative than they perhaps should be. Underdispersion is less of an issue as it just results in more conservatism in results.

Note, the individual tests that accompany the diagnostic plots tend to be stricter than the assumptions we are seeking to explore. That is, statistical models tend to be reasonably robust to mild assumption violations, yet the diagnostic tests are fairly strict. Hence, the tests are used to flag potential issues, yet the ownace is still on the researchers to explore these violations in greater detail and evaluate whether they are likely to have any important consequences.

Model Summaries

The coefficients (parameter estimates) table displays each of the main model parameter estimates on the scale of the link function. Parameters that start with `b_` are the population-level (fixed) effects. Parameters that start with `sd_` are the varying (random) effects.

The table lists the posterior median as well as lower and upper bounds of the 95% Highest Posterior Density (HPD) interval for each parameter. Also tabulated are the total number of posterior draws (`length`), Rhat values (`rhat`), and effective sample size in both the bulk of the posterior (`ess_bulk`) and tail of the posterior (`ess_tail`).

Analysis details

This panel has two sub-panels for displaying “Modelled trends” and “Modelled effects”.

Modelled trends

The modelled (predicted) values of a focal variable are displayed in tabular and graphical form.

In the table, each row represents the summaries of predictions for a specific year. The summaries are in the form of median as well as lower and upper 95% highest probability density (HPD) intervals of the full model posteriors. Years in which data included values below the limit of reporting are marked by a black flag (). The table also indicates which years are considered Baseline.

These same summaries are also presented graphically in the figure. Additionally, the figure displays the underlying un-modelled data in gray. Individual sites are connected by gray lines.

Modelled effects

This table summarises the comparisons between the average of the baseline year(s) and each of the subsequent reporting years. Values are reported on a **fold** (fractional) scale. On this scale, a value of 1 indicates no change. Values of 0.5 and 2 would indicate that the value in the reporting year is respectively half and twice that of the baseline year(s).

Posteriors of the comparisons are summarised by their medians as well as the lower and upper bounds of the 95% Highest Posterior Density (HPD) interval. Also tabulated are the exceedence probabilities associated with the probability that the values have declined since the baseline year(s) (`Pl`) and the probabilities that they have increased (`Pg`). These probabilities must sum to 1 and it is only necessary to explore the larger value per comparison. As a guide exeedence probabilities:

- **>0.95** provide strong evidence of an effect (change)
- **>0.9** provide evidence of an effect (change)
- **>0.85** provide week evidence of an effect (change)

Finally, there is also a column that represents the polarity and evidence of change via a color.

- red: strong evidence of an increase
- orange: evidence of an increase
- yellow: weak evidence of an increase
- white: no evidence of change
- light green: weak evidence of a decline
- mid green: evidence of a decline
- dark green: strong evidence of a decline

The full modelled posteriors are depicted in the figure under the table. The y-axis represents each of the contrasts (comparisons between sets of years or years and the Baseline). The posterior distributions are coloured according to the polarity and degree of evidence as outlined above.

The full and summarised posteriors can be presented on either a **percentage** scale (default) or **fold** scale and this is controlled via a dropdown box above the table.

- the percentage scale is interpreted as the percentage change between the contrasting years. For example, percentage changes of -20, 20 and 0 indicate a 20% decline, 20% increase and no change respectively.
- the fold scale is interpreted as the fold change between the contrasting years. For example, fold changes of 0.5, 2 and 1 indicate a halving, a doubling and no change respectively.

Bürkner, Paul-Christian. 2017. "brms: An R Package for Bayesian Multilevel Models Using Stan." *Journal of Statistical Software* 80 (1): 1–28. https://doi.org/10.18637/jss.v080.i01.

Gelman, A., and J. Hill. 2007. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge, UK: Cambridge University Press.

Hartig, Florian. 2022. *DHARMa: Residual Diagnostics for Hierarchical (Multi-Level / Mixed) Regression Models*. https://CRAN.R-project.org/package=DHARMa.

R Core Team. 2024. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.