# The Ocean Health Index Assessment Manual

## Contents

# 1 Introduction

**Summary:**

This guide provides an overview of conducting an OHI+ assessment. An assessment involves incorporating information from your study area into goal models to calculate Ocean Health Index (OHI) scores using the OHI Toolbox software and WebApp. This process is explained in this guide.

**The OHI framework** allows you to synthesize the information and priorities relevant to your local context and produce comparable scores. Because the methods of the framework are repeatable, transparent, quantitative, and goal-driven, the process of a carrying out an OHI+ assessment is as valuable as the final results.

The first completed assessment for a study area is valuable because it establishes a baseline and highlights the state of information quality and availability in an area. Any subsequent assessments carried out through time are also valuable because they can be used to track and monitor changes in ocean health. Your assessment will require careful thought and consideration along the way, and we encourage documentation and scripting

to be done within the OHI Toolbox to facilitate collaboration and transparency, as well as the reproducibility for future assessments.

Each OHI+ assessment should have a clear purpose. One of the typical reasons for conducting an independent assessment is to inform policy and management decisions. Assessments can be more relevant to management when they are conducted at the spatial scales at which policy decisions are made, such as states, provinces, or counties. The **regions** and the overall **study area** are definitions that will be used throughout the assessment. The study area is the entire spatial boundary of your assessment, while the regions are the smaller subdivisions within the study area. In the OHI framework, goal scores are calculated for regions separately and then combined to produce an overall OHI score for each study area. The number of regions varies with each assessment's study area; completed assessments have had between one and 221 regions.

> The process of conducting an OHI+ assessment is as valuable as the final results. Documenting decisions made, as well as the challenges and successes encountered along the way, can lead to better understanding of the system, help inform management decisions, and guide future assessments to track changes through time.

## 1.1 What to expect when conducting an assessment

> It is important to include information that best represents your study area, and to make science-driven decisions and clearly document what was done and why. Your team should as creative and insightful as you can be while working within the bounds of informational and technical limitations.

There are **key processes and considerations** that will be a part of every assessment. Every assessment should build from the lessons learned of previously completed assessments and identify what local characteristics need to be included in a study. This is done partly by comparing the local situation to situations in previous assessments; it is also done by comparing the default information provided in the WebApp to what is known about local realities. After you have outlined and identified local characteristics and priorities, you should prepare to use the Toolbox software and fit the data you have found to be formatted correctly for that software. Finally, once working with the information within the Toolbox, your team will udpate and improve the methods of the assessment before being able to calculate the final results. You will also visualize the outputs in the WebApp's maps and flower plots that can be shared with other partners and collaborators. Above all, you should be prepared to **know that this process takes time and is iterative, meaning that you often return to previous steps**.

How long does an assessment take? Past assessments have taken between two and three years, with the time varying depending the size and composition of the team, the challenges encountered in discovering and gathering information, and how many models are redeveloped. The amount of data processing and goal model development needed before you will be able to use the Toolbox also affects the amount of time it takes to conduct the assessment. The skill sets of the team members and the amount of technical resources available are also hugely important factors. You should think about which team members are needed at what stage of the process, including an R programmer and a spatial analyst. It will take time for the technical team to become familiar with the OHI Toolbox and GitHub.

## 1.2 Overview of best practices for OHI assessments

Conducting an assessment requires both an understanding of how past assessments have been completed and the innovation to capture important characteristics of your study area using the information available. You can start by understanding the structure of completed assessments at global and smaller scales and the models that were created. Understanding the approaches in different contexts will help you think about what should be done similarly and differently in your local context.

Navigating through the WebApp can help frame your thinking and introduce you to the structure of inputs that will be required for the OHI Toolbox software.

### 1.2.1 Understand completed OHI assessments

It is important to **understand methods used in completed OHI assessments** so that you can identify if previous approaches are appropriate for your assessment or whether you need to redevelop the methods for your study area. In many cases, studying completed OHI assessments will inform your approaches for discovering data and developing goal models later on in the process.

The OHI framework was developed through collaboration and iteration. Your assessment can leverage the collective knowledge and insight used in the methods of the global assessment by Halpern *et al.* in *Nature* (2012) as well as the subsequent assessments conducted annually (in 2013, 2014, and ongoing). Each annual global assessment has improved upon some of the goal models based on better data availability or a better understanding of the systems involved. Several smaller-scale assessments have been completed that are highly informative as well, and particularly for regional scale assessments. The following studies have been published with supplemental online materials, and are available at http://ohi-science.org:

- **Global**
- Halpern et al. (2012) An index to assess the health and benefits of the global ocean. *Nature*.
- Halpern et al. (2015) Patterns and emerging trends in global ocean health. *PLoS ONE*.
- **Brazil**
- Elfes et al. (2014) A regional-scale Ocean Health Index for Brazil. *PLoS ONE*.
- **United States West Coast**
- Halpern et al. (2014) Assessing the health of the U.S. West Coast with a regional-scale application of the Ocean Health Index. *PLoS ONE*.
- **Fiji**
- Selig et al. (2015) Measuring indicators of ocean health for an island nation: The Ocean Health Index for Fiji. *Ecosystem Services*

Additionally, several OHI+ assessments have been completed. As information is available about those assessments they will be posted on http://ohi-science.org.

> TIP: The OHI+ development team is prepared to provide guidance for assessments.

### 1.2.2 Incorporate core values and characteristics into the OHI assessment framework before gathering information

**Begin your assessment by identifying local socio-cultural-economic characteristics and priorities related to ocean health, and how they would ideally be captured with the existing or modified OHI framework**. This means understanding the rationale behind the components of the OHI framework and identifying what must be added or removed or redefined to ensure that it best represents the local context. Are all goals relevant to your study area? What should be added, removed, or redefined? In this process it is important to identify not only characteristics that could be included in goal models, but also the important stressors (pressures) and resilience elements within the study area. What are the key issues that should be included for your assessment to be credible, useful, and meaningful? How do people typically relate to the ocean in your area in terms of social and cultural patterns? These are the kinds of questions you should consider prior to assembling the available information.

The OHI framework should guide your assessment, but you should not be constrained by it. If a goal is not relevant, it should be removed. If there are elements important to your study area that are not present within the existing framework, how could they be included? Having a clear picture of how the framework should be restructured and what the assessment should include is very important before moving on to assemble information, because otherwise the assessment could be biased by what information is available instead of what is important to include. When specific information is not available there are ways to capture them with indirect measures, called proxies, which will be discussed in the **Assemble Input Information** section.

#### 1.2.3 Strategically define spatial boundaries at the finest possible scale

Identifying the spatial boundaries of the regions within the assessment area is extremely important because OHI scores are calculated for each unique region. Spatial boundaries should be defined with geographic information system (GIS) mapping software at the smallest scale possible, ideally within one management jurisdiction. This is optimal because it is often at these scales where management and policy decisions are made, cultural priorities and management targets are identified, and information is collected in standardized and therefore comparable ways.

#### 1.2.4 Assemble input information

There are many decisions to be made when searching for and gathering data, and searches should extend beyond any one expertise, discipline, source, or data-type. This is because your data will come from disparate sources, and you will have to engage experts to help identifying good proxies and indicators, deciding reference points, and developing goal models. OHI+ assessments should incorporate higher-resolution information where possible for goal status models and pressures and resilience measures. The process of discovering and gathering so many different kinds of data and indicators is an important step that you will return to as you continue to conduct the assessment. This is where having a collaborative team that can work across disciplines will be key.

#### 1.2.5 Maintain core values and characteristics within the assessment framework regardless of limited information quality

The models you develop and reference points you set must reflect the philosophy of the OHI framework while accommodating the attributes and shortcomings of the data. While goal status models developed in completed assessments offer good examples of approaches in different contexts, they should be considered as guides and should not limit exploration into new model development. It will likely be an iterative process to incorporate the best available information into a model that captures the philosophy of the goals. It will also require creative thinking and problem-solving abilities among your team, and documenting the decisions you make is important for transparency, communication and repeatability.

#### 1.2.6 Carefully document and share all decisions in writing and computational code

It is important to plan for future assessments, as repeated assessments enable you to compare and track how scores have changed over time, with the aim of ultimately informing policy to improve ocean health. Repeated assessments will use the same methods and reference points, but incorporating updated data.

Detailed information about how the assessment is conducted will enhance its credibility and reproducibility. Decisions of why information was included and why models were developed in a certain manner are of great importance so that future assessments can incorporate the same logic and understanding of the system — or make improvements. Further, the type of workflow developed and software used to organize and process information will greatly affect the efficiency, transparency, and reproducibility of subsequent assessments. Providing public access to all such information, as well as input data and computational code is becoming the standard for scientific inquiry, so every effort should be made to achieve those aims.

### 1.3 The Toolbox software and WebApp

The **OHI Toolbox** is the main software used for organizing and processing information, documenting decisions made, calculating scores, and visualizing results. It was created to facilitate score calculations as well as the organization of information and transparency of the approach. The Toolbox is built with open-source software, meaning that it is freely available for you to use and can be modified to meet your needs. You

will access the software from online and will use several free software tools to conduct your assessment in a collaborative, transparent, and reproducible manner.

The Toolbox software should be used when your team has gathered the information necessary for the goal models you have developed. The steps of carefully preparing and gathering data layers, indicators, and developing goal models can be done independently of the actual software use; however, if done within the Toolbox structure these steps will be traceable and collaborative. But of course the assessment can be done without all members of your team becoming familiar with the technical aspects of the Toolbox.

Accompanying the OHI Toolbox is the **OHI+ WebApp**, which is a visualization tool that displays input information and calculated scores in several ways, including interactive maps and flower plots. Most coastal countries have a WebApp that was created to facilitate planning and communication during assessments. The WebApp visually presents inputs, goal models and calculated scores for each defined region through interactive maps, histograms, and tables. All inputs presented in a country's WebApps were extracted for each country from global assessments, and scores were calculated using goal models from global assessments.

The WebApp is a widely useful communication tool. Not all team collaborators may be involved with the technical aspects of the Toolbox, but the WebApp enables everyone to explore inputs and calculated scores.

### 1.4 Outcomes of conducting an assessment

**The process of conducting an OHI assessment can be as valuable as the final results.** This is because while conducting an OHI assessment you will bring together meaningful ocean health information from many disciplines. In doing so, you will have a census of existing information and will also identify knowledge and data gaps. Further, conducting an OHI+ assessment can engage many different groups, including research institutions, government agencies, policy groups, non-governmental organizations, and both the civil and private sectors.

Your completed assessment will produce OHI scores for each goal for every region in your study area, and scores within the assessment can be compared with each other. These scores will not be quantitatively comparable to those of other OHI assessments because they differ in the underlying inputs, goal models, and reference points. The only *quantitative* comparisons can be made within an assessment's study area, whether between regions or through time. However, *qualitative* comparisons between different OHI assessments can be made because the scores are an indication of how far a region is to achieving its own targets. For instance, if two study areas have scores of seventy and sixty-five, it should be interpreted that the first study area is closer to its management targets than the second is, but since these management targets are different (in addition to the underlying data and models), they cannot be quantitatively compared.

## 2 Overview of the OHI WebApp

**Section Summary:**

Your team should be familiar with the structure of the WebApp since it demonstrates how information is organized and displayed. As you update inputs with local information, you can view these updates with the WebApp.

**OHI+ WebApps** are websites created to facilitate independent assessments. The WebApp is a good starting point when conducting an assessment because you can easily navigate how information is organized and displayed. The WebApp is also meant to be used to visualize and communicate results.

The WebApp displays input information (data and indicators) as well as final OHI scores. When prepared and formatted for the OHI, inputs are called **layers** and are used in all OHI calculations, including goal models, pressures and resilience. By default, the WebApp only displays layers and score information that have been extracted from the latest global assessment and allocated to subcountry regions with the study

area. The default display therefore does not provide fine resolution nor does it guarantee accurate data for each study area. You will substitute these layers with higher-quality information at the local scale in your assessment. However, the default layers can be used as inputs into your assessment in cases where no better information exists. **Incorporating the best information possible will generate results that best represent your study area.**

The WebApp is powered by the **OHI Toolbox**, which organizes all of the layers and calculates Index scores. The Toolbox is where you will actively work to prepare and format layers and develop goal models, which can then be displayed with the WebApp.

A default WebApp is available for most coastal nations. For example, Ecuador's WebApp (ECU) is found at http://ohi-science.org/ecu. Note that it is possible to translate the page into your language of choice.

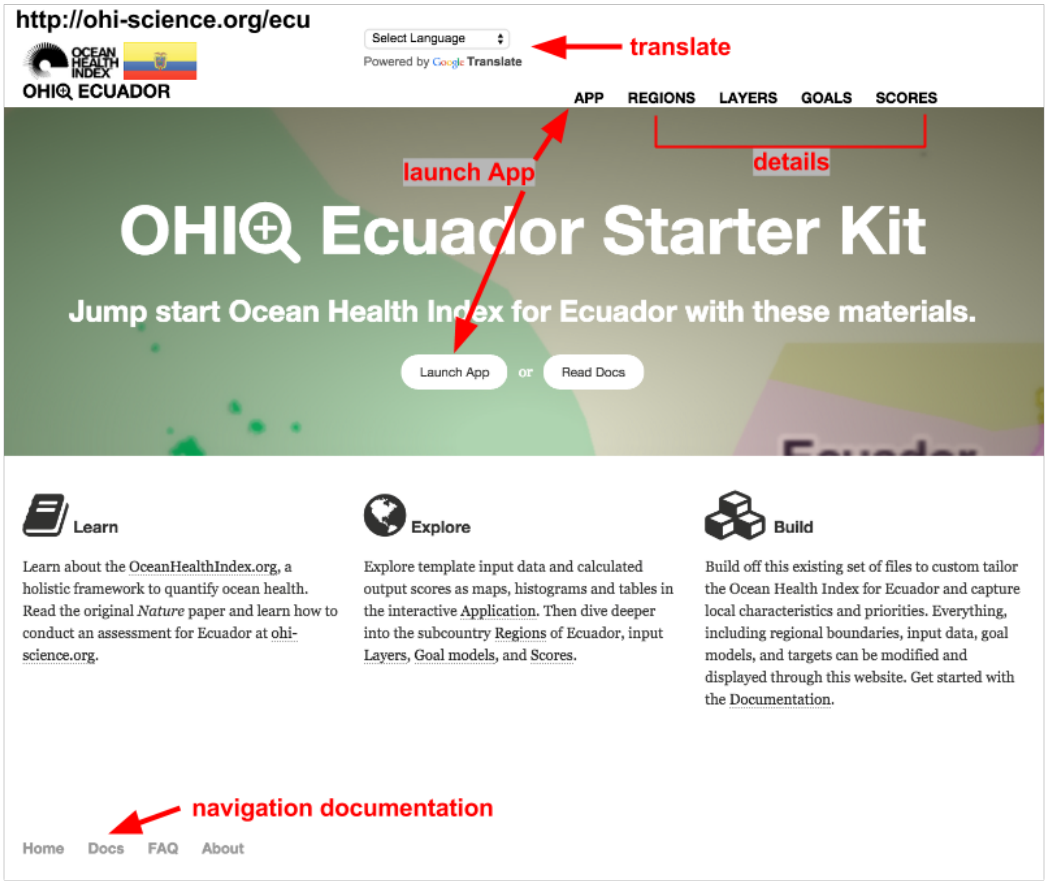



Figure 1: The WebApp start page. Note that it is possible to translate the page into your language of choice.

> *Remember that this information is publicly available when stored on free GitHub accounts. If you would like your assessment to be private, you can choose a premium option.*

The WebApp homepage provides tabs for you to explore your data, regions, and calculated scores. The main pages are **App**, **Regions**, **Layers**, **Goals**, and **Scores**, which were described in the **Conceptual Guide**. The App page is described more below.

The interactive **App** page allows you to explore input and output variables. The inputs are the layers and the outputs are calculated scores for each goal and dimension of the Index for each region in the study area.

This page is where you should start your exploration. By choosing input layers you can see the range of values for a given variable and you can also see information about data sources. More detailed information about the default goal models and the calculation methods is found in the other pages of the WebApp.

The information displayed on the website is stored online in a **GitHub repository.** GitHub is an open-source development platform that allows multiple users to collaborate, track changes, and share their work to prepare data files and write code. Some members of your team will use GitHub to track layer preparation and view the history of changes made in this process. It also a a way for your team to document the decisions made during your assessment. Any changes made to files contained within the GitHub repository will be automatically displayed on the WebApp for other team members and collaborators to view. The history of these changes is also stored on this platform as an archive, and it can be used to display changes made over time (See the section on **GitHub** for how to modify files using this platform).

## 2.1 Defining and displaying regions

WebApps display subcountry regions within each study area. The boundaries for these subcountry regions are usually states, provinces, or districts reported to Global Administrative Areas (GADM: www.gadm.org). These land-based regions are extended offshore to divide the Exclusive Economic Zones (EEZs) into offshore regions of the study area. Offshore regions are important for Index calculations, in part because scores for each region are combined using the offshore area to weight the average of the final Index score. You can redefine these regional boundaries; these subcountry regions have been provided as a starting point. To redefine the boundaries you will need a spatial analyst; details are below.

It is important to note that the provided WebApps do not claim to take a stance on disputed territories. The boundaries for all EEZs were identified by MarineRegions.org (http://www.marineregions.org), and subcountry regions were identified by the Global Administrative Regions database (http://gadm.org).

## 2.2 Exploring inputs and outputs with the WebApp's App page

The App page allows you to explore the input layers and calculated output scores for each region in the study area. The page presents data and scores from the global assessment that are applied to each subcountry region in the study area by default. In order to explore data and scores, you can select them on the left to view their attributes and you can also visualize them in a number of ways on this page.

The App page allows you to view displays through the **Data** and **Compare** tabs. The **Data tab** provides summary information on each layer and metadata descriptions that accompany them. The **Compare tab** is mainly used for comparing output scores when modifications are made to the underlying data or models.

### 2.2.1 The App's Data tab

**2.2.1.1 Overview of display options** The Data tab displays input layer or calculated scores for each goal parameter. It presents the information through a *Map*, *Histogram*, or *Table*. These options are available as sub-tabs on the Data tab page. The *Map* sub-tab is the default display option for the Data tab, and all data presented are drawn from the global assessments by default. This means they are either directly duplicated across regions, or the raw values are down-scaled using offshore area- or population-weightings. The *Histogram* sub-tab likewise draws from the same data source, but it displays a histogram of observed values with a smoothed line added. The *Table* sub-tab also draws from the same data but offers information in a table.

**Data displayed in the Map sub-tab:**

The *Map* displays data for every region. A legend is displayed in the lower right-hand corner of the map to explain the meaning of the colors presented. The range of values will change when variables are selected, and the colors will automatically change to match that range.

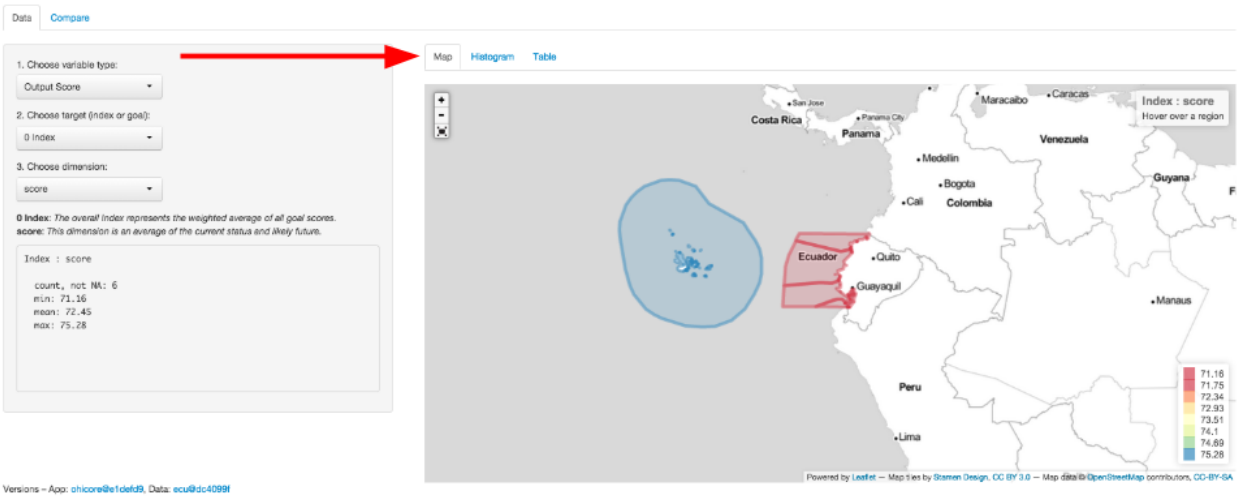



Figure 2: The Map sub-tab. Click on 'Map' to see a geographic view of your assessment region. Colors indicate scores or values for your input layers or output scores. This example shows Index scores for each region in Ecuador.


Figure 3: The Histogram sub-tab. Click on 'Histogram' to see the distribution of layers or scores after selecting a variable layer on the left. This example shows the Species sub-goal scores for the study regions of Ecuador.

**Data displayed in the Histogram sub-tab:**

The *Histogram* shows the distribution of values of the selected variable as the number of observations for each value bin (shown as white bars) and it also automatically creates a smoothed density function (shown as pink shading).

**Data displayed in the Table sub-tab:**

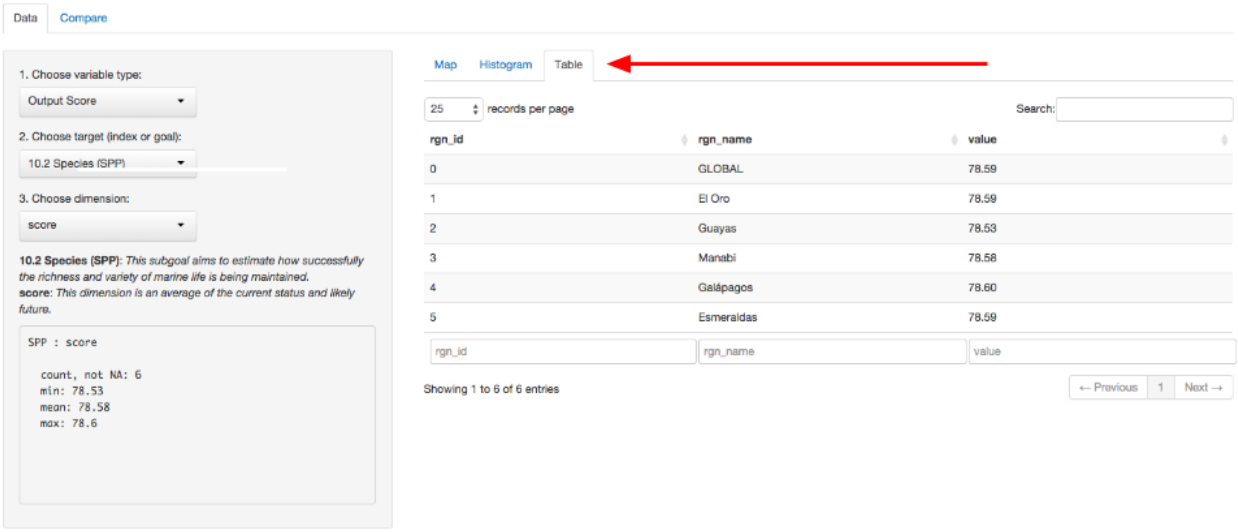



Figure 4: The Table sub-tab. Click on 'Table' to see a table of your data or scores, after selecting a variable layer on the left. This example shows the Species sub-goal scores for the regions of Ecuador.

The *Table* displays the variable's value for each region in the study area. It provides an identifying code (*rgn_id*), name (*rgn_name*), and value (*value*) for each observation. The variables are also searchable since you can use the textbox at the bottom of the page to filter the results displayed.

**2.2.1.2 Overview of variable options** When you choose the variables to be displayed, you will also see summary descriptions for the layer. These descriptions, statistics, and metadata for the chosen fields are displayed below the drop-down menus on the left side of the page.

TIP: As you prepare new layers, your updated descriptions will appear here.

The first selection you should make from the drop-down menus is the variable type. This means you can choose either **Input Layer** or **Output Score**. The **Input Layer** will show the layer used for a particular target you select. The targets in this case are either goals, pressures, resilience, or spatial information. The **Output Score** will show calculated scores for the alternative target selections you will make. In this case, the targets are Index or goal scores. In either case, you then have the option to further refine your search by either going into a specific layer or a specific dimension that is used in the overall Index calculations. If you do not make a selection, the **Output Score** is displayed by default.

For example, if you select **Output Score** as the variable type, you will then be able to choose a target goal or sub-goal, and then you will be able to choose the OHI dimension to be displayed. Remember that the dimensions are status, trend, pressures, resilience, future state, and score. In this way you can investigate the components that combine to create the goal scores.

As another example, if you select **Input Layer** as the variable type, you will be able to choose a target such as a goal and a specific layer associated with that goal. If that input layer has multiple categories of input types, or if it has multiple years of information available, you will be able to select more specific information.

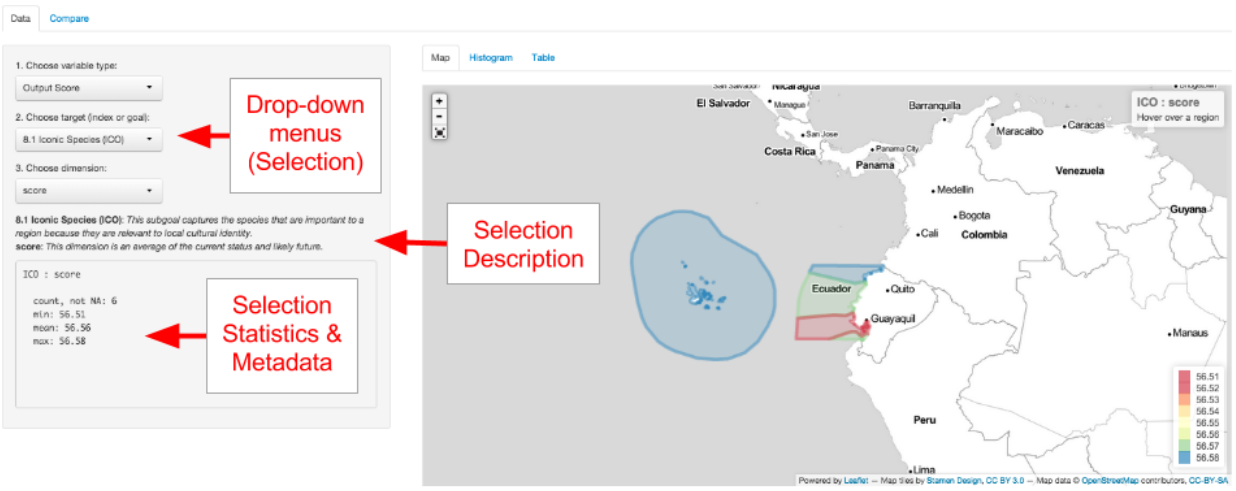



Figure 5: Overview of the Data tab. Choose the variable you would like to explore through the drop-down menus on the left-hand side of the page. Once you select either input data or an output score, you can view a description and summary of values below.

If you do not make a selection, the default setting is the first alphabetical category and the most recent year available.

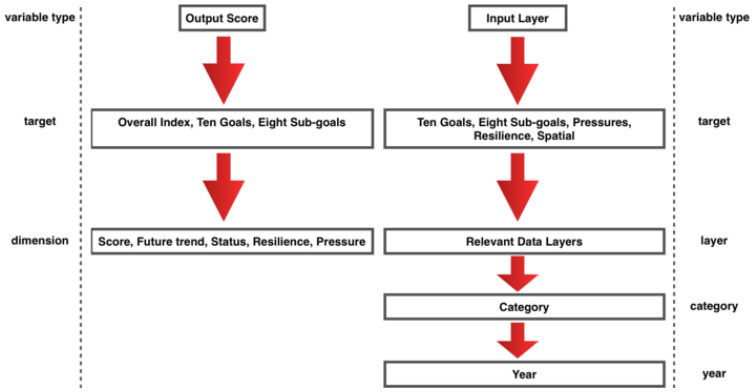



Figure 6: Overview of the variable options. You can choose to select and view either `input layers` or `output scores`.

### 2.2.2 The App's Compare tab

The **Compare** tab allows you to compare differences in calculated scores based on changes you have made to the underlying layers. These changes can be the values of the layers themselves, or they can be from changing the goal models. Any component you change is tracked through the archiving system of **GitHub** (See the section on **GitHub**) and each version of the changes be visualized here. You can take advantage of this ability to compare one saved version of your calculated Index output to another version of your calculated output. This is done to compare the how changes made to your data or indicators for goals, pressures, or resilience would affect the resulting scores.

You can use this in two ways. One way is use this is to visualize updates as you make them: viewing differences is extremely helpful for error checking and for sharing tests to the data among your technical team. Another way to use the Compare tab is to compare different management scenario to how changes in your assumptions would impact score results. These changes could occur in the goal models themselves, such

as through changes to targets or reference points, or they could be changes made to the values of pressures and resilience layers, for example.

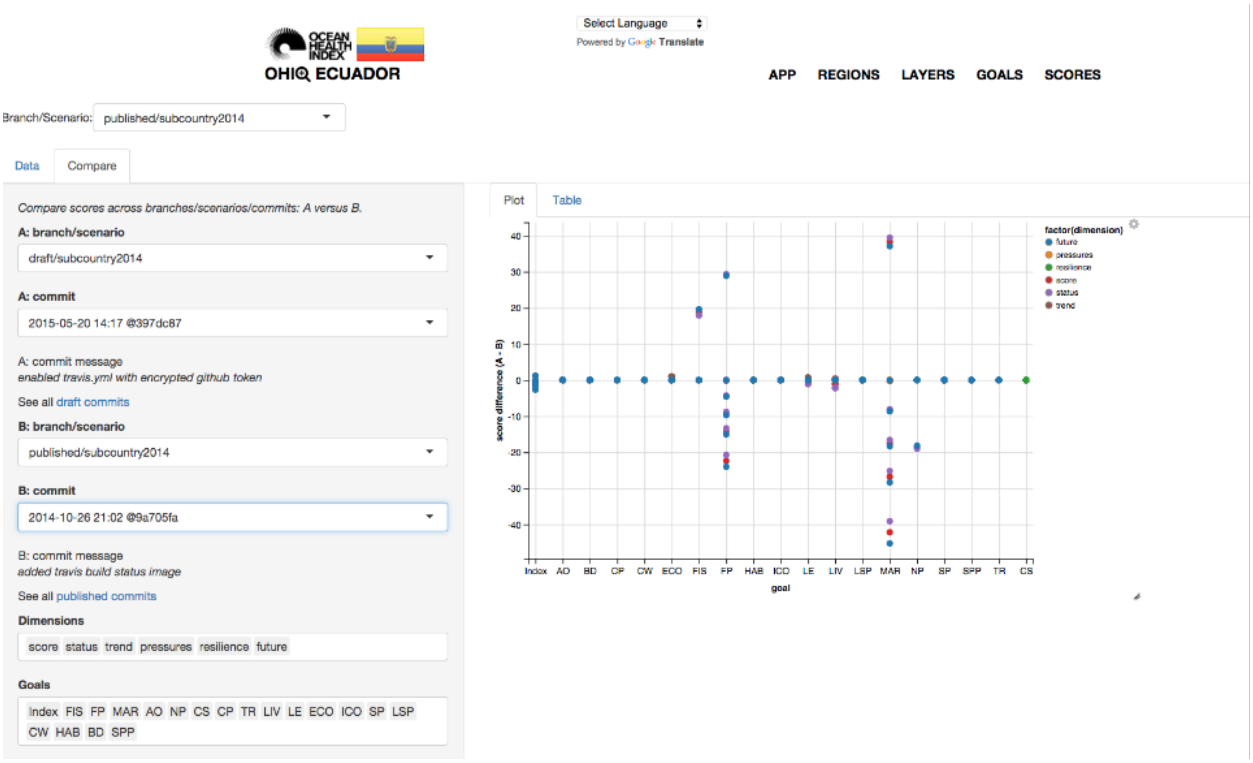



Figure 7: You can use the OHI+ WebApp's 'Compare' tab to error check, and check the outcomes of alternate scenarios of your assessment.

The App page also offers the ability to view different **branches** or **scenarios** in the upper left-hand corner of the page. The **branches** refer to unique copies of a GitHub repository where information is stored. Each branch is a copy of a repository that is meant to be modified independently of other branches. This is done to ensure that changes made to one branch will not affect the information in another branch. This allows for active collaboration and offers a way to archive different outputs to Index calculations. For example, the *published* branch shows information that has been vetted for sharing, while the *draft* branch can be used for experimentation. These branches can be merged together at any time, and that is typically done when important milestones in the assessment process are reached. The *subcountry* folders displayed also offer another way to compartmentalize outcomes by allowing you to compare different *scenarios* within the same branch of your repository.

> The App displays a *published* branch by default. It is recommended work on the *draft* branch until your assessment is finalized. When it is finalized, you can then merge the *draft* branch with the *published* branch.

These options for displaying and comparing information will be useful for understanding the multiple objectives in your OHI+ assessment.

# 3 Defining spatial boundaries

Defining spatial boundaries for the reporting regions (study area and regions) is a very important step in the assessment process. It is important because all data, analyses, and results will be at this spatial scale, and

boundaries may be used to aggregate or disaggregate information reported at spatial scales different from your regions. Boundary definitions should match the purpose of the assessment and be informed by the scale at which information is available.

**It is possible to redefine the spatial boundaries for your study area and regions.** The boundaries displayed in your WebApp** are provided by default using subcountry region definitions from Global Administrative Areas (GADM: www.gadm.org).

> Note that the OHI does not take a stance on disputed territories. The boundaries are defined by the original map data providers.

There is no limit to the number of regions that can exist within the study area; the size and number are only constrained by data availability and the utility of having scores calculated for a particular region. Although it is possible to assess only one region in the study area (i.e. the region is the assessment area), this might not be ideal because it eliminates the possibility of making comparisons or identifying geographic priorities within the study area.

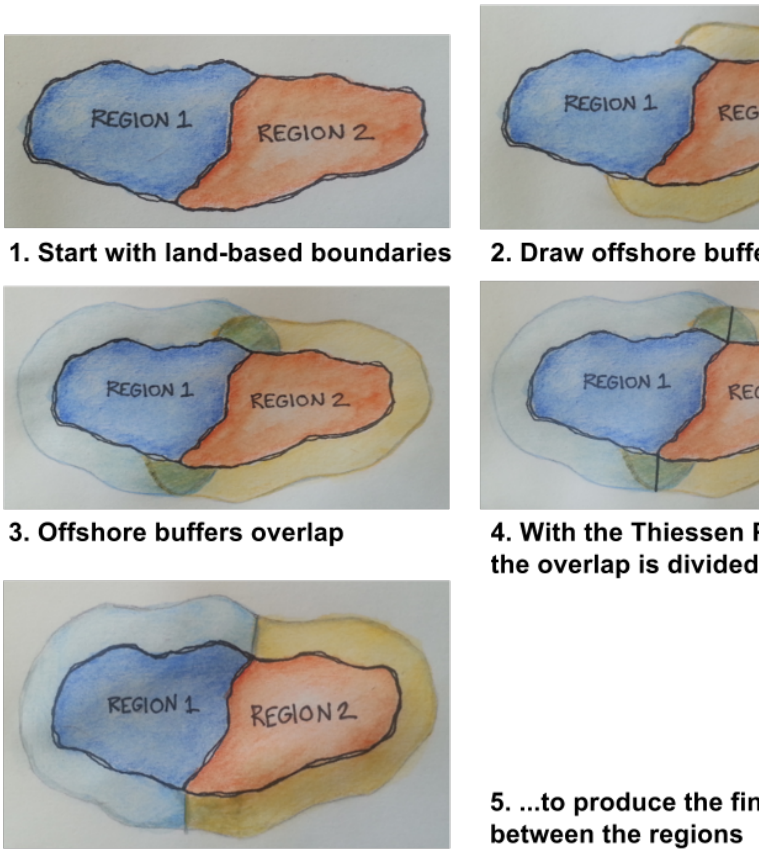
## 3.1 Drawing spatial boundaries

**Spatial boundaries must be drawn with geographic information system (GIS) mapping software** such as ArcGIS, QGIS, or GRASS. You will need someone with GIS skills to create a **shapefile** that will be used by the Toolbox to display your information. The shapefile will also be used to extract information for each of your defined regions when data are reported in raster format for a different area. For more information see https://en.wikipedia.org/wiki/Geographic_information_system and http://en.wikipedia.org/wiki/Shapefile.

Regions must be unique (non-overlapping), and boundaries must be drawn offshore, extending to the exclusive economic zone (EEZ) edge in most cases. Offshore boundaries should be made with spatial methods in order to extend boundaries from those designated on land. One possible method is to create boundaries with Thiessen Polygons, and we provide a Python script that can be used, but it requires ArcGIS. The Python script and further details can be found at http://ohi-science.org/pages/create_regions.html. Using Thiessen Polygons, offshore boundaries are created with the following steps.

1. Start with land-based boundaries
2. Draw offshore buffers for each region

3. But the buffers overlap
4. For the Thiessen Polygon approach, the overlap is divided




5. To produce the borders between the regions

#### 3.1.1 Updating the map in your WebApp

**Once you have created your boundaries with GIS software, you will need to send them to us.** Please send us a .zip file of all files produced. Files with the following extensions are required (but you can send all files):

- `.dbf`
- `.shp`
- `.shx`
- `.prj`

The `.dbf` file needs the following in its attribute table:

- **rgn_id** (unique numeric region identifier)
- **rgn_name** (unique named region identifier)
- **area_km2** or **area_hectare** (area in km2 or hectares)

Note that we can send you the shapefiles used to create your WebApp if you would like to modify them.

#### 3.1.2 Buffers

When drawing your regions, it is also a good idea to create inland and offshore buffers that will be used to extract data in your assessment. Buffers are not necessary for display in the WebApp but they will be

important for later layer preparation. For example, the global assessment used coastal population information, and raster data were available for entire countries. This meant that 'coastal' had to be defined: for global assessments it was defined as 25 miles from the coast. To extract just the coastal population from the population raster file, we created a 25 mile inland buffer for each reporting region. But to extract mangrove data for each region from raster files, global assessments used 1km inland and 1km offshore as the buffer.

At this point, you may not know which buffers you will need, as they depend on the data available, your goal models and definitions. Some buffers used in the global assessments were 1km inland, 25miles inland, 1km offshore, 3nm offshore.

# 4 Discovering and gathering input information

A hallmark of the OHI is that it uses freely-available existing information (data and indicators) to create the models that capture the philosophies of individual goals. The quality of the inputs are important because calculated Index scores area only as good ad the inputs on which they are based. Assembling the appropriate input information, which means both discovering and gathering data and indicators, is an important part of any OHI assessment.

Once your team has tailored the OHI framework appropriately for your study area and identified the information that ideally would be included, the data discovery and gathering process can begin. There are many decisions to make when deciding which data are available and appropriate to include in your assessment. Finding appropriate data requires problem-solving abilities and creativity, particularly when ideal data are unavailable. You will need input information to calculate status models as well as pressures and resilience.

## 4.1 Thinking creatively

> Remember that you are trying to capture information that is meaningful for ocean health.

Humans interact with and depend upon the oceans in complex ways, some of which are easy to measure and others of which are harder to define. More familiar measurements include providing seafood, or disposing of waste. A less familiar measurement is how marine-related jobs affect coastal communities, or how different people receive or perceive benefits simply from living near the ocean. Thinking creatively and exploring the information available can make your assessment more representative of reality.

Data used in OHI assessments spans a wide array of disciplines beyond oceanography and marine ecology. It is important to think creatively and beyond the interests of a specific institution or one particular field of study. Therefore, it is necessary to look beyond the most known or obvious data sources to find data relevant for the goals in the study area. Discussions with colleagues, literature searches, emails to experts, and search engines are good ways to understand what kinds of data are collected and to hunt for appropriate data. Investigate what kinds of information are available from government and public records, scientific literature, academic studies, surveys and reports etc.

## 4.2 Data sources

Existing data and indicators can be gathered from many sources across environmental, social, and economic disciplines. This includes government reports and project websites, peer-reviewed literature, masters and PhD theses, university websites, and information from non-profit organizations, among others.

All data must be rescaled to specific reference points (targets) before being combined with the Toolbox; therefore setting these reference points at the appropriate scale is a fundamental component of any OHI assessment. This requires your assessment team to capture the philosophy of each Index goal and sub-goal using the best available data and indicators. Indicators that are already scaled (e.g., from 0-1 or 0-10) can easily be incorporated into your assessment; they have already been scaled to some kind of identified target

or reference point. Data that have not been scaled in most cases will need to be, whether this is by scaling to the maximum value in the range or to some other understood value. You should think about how you would rescale data during your search.

Because data and indicators will come from different sources, they will also have different formatting. To include these data and indicators in your assessment, you will need to process these files into the format required by the Toolbox, which is explained in the section **Formatting Data for the Toolbox**. When data have been prepared and formatted for the Toolbox, they are called **layers.** Because creating layers can be quite time-intensive, data should only be prepared for the Toolbox after final decisions have been made to include the data or indicator in your assessment, and after the appropriate goal model and reference points have been finalized.

## 4.3 Gathering responsibilities

Gathering appropriate data requires identifying and accessing existing data. It is important that team members responsible for data discovery make thoughtful decisions about whether data are appropriate for the assessment. Data discovery and acquisition are typically an iterative process, as there are both practical and philosophical reasons for including or excluding data.

## 4.4 Requirements for data and indicators

There are six requirements to remember when investigating (or 'scoping') potential data and indicators that are presented in this section. It is important that data satisfy as many of these requirements as possible. To meet these requirements, you may have use appropriate methods to fill gaps in the data set. Data sources may need to be excluded from the analyses if requirements are not met and gap-filling solutions are not possible. If data cannot be included, you may elect to use layers from the global assessment or identify other data or modeling approaches.

### 4.4.1 Relevance to ocean health

There must be a clear connection between the data and ocean health, and determining this will be closely linked to each goal model.

### 4.4.2 Accessibility

The two main points regarding accessibility are whether the source is open access and whether the data or indicators will be updated regularly.

The Index was created in the spirit of transparency and open-access, using open-source software and online platforms such as GitHub, is to ensure as much accessibility and open collaboration as possible. Data and indicators included should also follow these guidelines, so that anyone wishing to understand more about the Index may be able to see what data were used and how. For this reason we emphasize the importance of using data that may be made freely downloadable, as well as the importance of clearly documenting all decisions and reasons for the choices made in selecting data, indicators, and models.

Index scores can be recalculated annually as new data become available. This can establish a baseline of ocean health and serve as a monitoring mechanism to evaluate the effectiveness of actions and policies in improving the status of overall ocean health. This is good to keep in mind while looking for data: will it be available again in the future? It is also important to document the sources of all data so that it is both transparent where it came from and you will be able to find it in the future.

#### 4.4.3 Quality

Understanding how the data or indicators were collected or created is important. Are they collected by a respected organization with quality control? Are there any protocol changes to be aware of? For instance, were there changes in the collection protocol to be aware of when interpreting temporal trends?

#### 4.4.4 Reference point

Most data will need to be scaled to a reference point. As you consider different data sources it is important to think about or identify what a reasonable reference point may be. Ask the following types of questions as you explore data possibilities:

- Has past research identified potential targets for these data?
- For example, fisheries goal require a Maximum Sustainable Yield (MSY).
- Have policy targets been set regarding these data?
- For example, maximum levels of pollutants allowed in beaches.
- Would a historic reference point be an appropriate target?
- For example, the percent of habitat coverage before coastal development took place.
- Could a region within the study area be set as a spatial reference point?
- For example, a certain region is regarded as the leader in creating protected areas.

#### 4.4.5 Appropriate spatial scale

Data must be available for every region within the study area. It is not always possible to fully meet the spatial and temporal requirements with each source. In these cases, provided that the gaps are not extensive, it can still be possible to use these data if appropriate gap-filling techniques are used (See: **Formatting Data for Toolbox** section).

#### 4.4.6 Appropriate temporal scale

Data must be available for ideally the five most-recent years to calculate the recent trend. For some goals, where temporal reference points are desirable, longer time series are preferable.

#### 4.4.7 The process of information discovery

The most important thing to remember when gathering data and indicators is that they must contribute to measuring ocean health. Not all information that enhances our knowledge of marine processes directly convey information about ocean health and may not be appropriate within the OHI framework. Because of this, compiled indicators can sometimes be more suitable than raw data measuring single marine attributes.

Whether you are working goal-by-goal, or layer by layer, it is important to consider where you can find synergies in data discovery. For example, while you are looking for information for the fisheries goal, you may also find data layers for fishing pressures, such as metrics on bycatch or trawling intensity. This will save you time and allow you to start thinking about how to rank pressures and resilience weights on your goals as well. Conceptually, it will help your team build a picture of how your goals are interlocking in a way that is reflective of the actual linkages that exist in the connected systems you are studying. Some key examples are listed below, and are further explained in the following sections.

You should begin by understanding and comparing the best approaches used in assessments that have been completed, including the global assessments (Halpern *et al*, 2012; 2013), Brazil (Elfes *et al.* 2014), Fiji (Selig *et al.*, 2014), and the U.S. West Coast Assessment (Halpern *et al.*, 2014). For OHI+ assessments, if

finer-resolution local data were available in the study area, these data were either incorporated into modified goal models that used locally appropriate and informed approaches or into the existing global goal model. When local data were not available, the global-scale data and global goal models were used, which is least desirable because it does not provide more information than the global study.

When looking for data, the following decision tree may be useful when going goal-by-goal for discovering data and developing models:

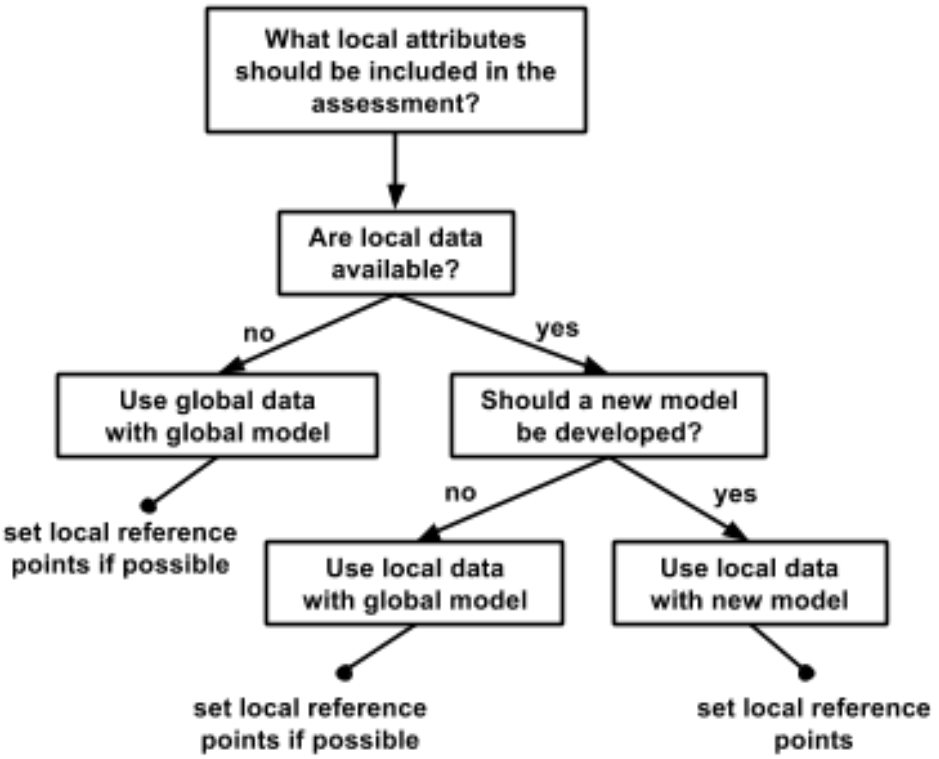



Figure 8:

#### 4.4.8 Example: U.S. West Coast data discovery

Below are examples of some decisions made when exploring available data for the U.S. West Coast assessment. Determining whether certain data could be included began with a solid understanding of the layers and models included in the global assessment. Since the U.S. West Coast is a data-rich region, finer-resolution local data could be used in place of many of the global data layers. The U.S. West Coast assessment had five regions: Washington, Oregon, Northern California, Central California, and Southern California.

#### 4.4.9 Reasons data were excluded

There are a lot of existing data that contribute to our scientific understanding of ocean processes and interactions but are not ideal for the OHI. Reasons to exclude data are both due to practical requirements (e.g., spatial or temporal resolution) and philosophical requirements (i.e., they do not help capture the attributes of interest for assessing ocean health). Some common reasons for excluding data are:

- **The data do not cover the entire area of the reporting region**. The state of California had excellent, long-term data on public attendance at state parks that would have been quite useful in the calculation of the tourism and recreation goal. However, data were only available for three of the five regions (the three California regions but not Oregon and Washington), so they could not be used.
- **There is not a clear and scientifically observed relationship between the data and ocean health**. Along the U.S. West Coast, kelp beds are a very important habitat because of their contribution to biodiversity and coastal protection. However, kelp coverage variation is driven primarily by abiotic natural forcing (wave or storm disturbance and temperature) and thus is not a good indicator of kelp forest health, particularly in the case of anthropogenic impacts. For these reasons kelp coverage was not included in the assessment.
- **The feature being measured may provide benefits to people, but this feature is not derived from marine or coastal ecosystems**. Sea walls and riprap provide coastal protection to many people along the U.S. West Coast. However, these structures are not a benefit that is derived from the marine ecosystems, so only coastal habitats were included in the calculation of this goal. These data can be included as a pressure due to habitat loss. They were not used as a resilience measure because they can often have negative side effects (e.g., by altering sedimentation dynamics), and because they have limited long-term sustainability (i.e., they need maintenance).
- **Data collection is biased and might misrepresent ocean health**. The U.S. Endangered Species Act identifies a species list focused on species of concern within the US. As such, these data are biased in the context of ocean health since they only assess species whose populations may be in danger. For the calculation of the biodiversity goal, using these data would be inappropriate because this goal represents the status of all species in the region, not just those that are currently of conservation concern. Using these data may have shown the status of biodiversity to be lower than it really is because the selection of species to assess was already biased towards species of concern.
- **Time series data are not long enough to calculate a trend or a reference point** (when a historical reference point is most appropriate). For the U.S. West Coast, the current extent of seagrass habitats was available, however, these do not exist for previous points in time in most areas, so could not be used to calculate the trend or set a historical reference point. Therefore, we estimated the trend in health of seagrass habitats using as a proxy the trend in the main stressor (i.e., turbidity). In other words, we assumed that the rate of seagrass loss was directly proportional to the rate of increase in turbidity. Similar solutions may be used to estimate trends in your own assessment, if there is scientific support for assuming that the trend of what we want to assess (or the relationship between the current state and the state in the reference year) has a strong relationship with the trend of the proxy data available.

# 5 The Ocean Health Index Toolbox

**Section Summary:**

In this section, you will learn the basics of how to use the OHI Toolbox to conduct your assessment. You will be introduced to files the Toolbox requires, how you will modify them, and how they interact to calculate the final output scores.

**The OHI Toolbox** is an ecosystem of small files and scripts that are the tools needed to calculate OHI scores. These files and scripts are stored in two '*repositories*': folders that are synchronized with collaborators. The first folder is your **assessment repository** that has a three-letter code, such as *esp* for Spain or *ecu* for Ecuador. You will edit this repository with your data, goal models, and updated pressures and resilience matrix tables. The second repository is called **ohicore** and it contains core functions for combining your data and goal models to calculate OHI scores. You will not edit `ohicore`, but you are able to explore it to understand the calculations.

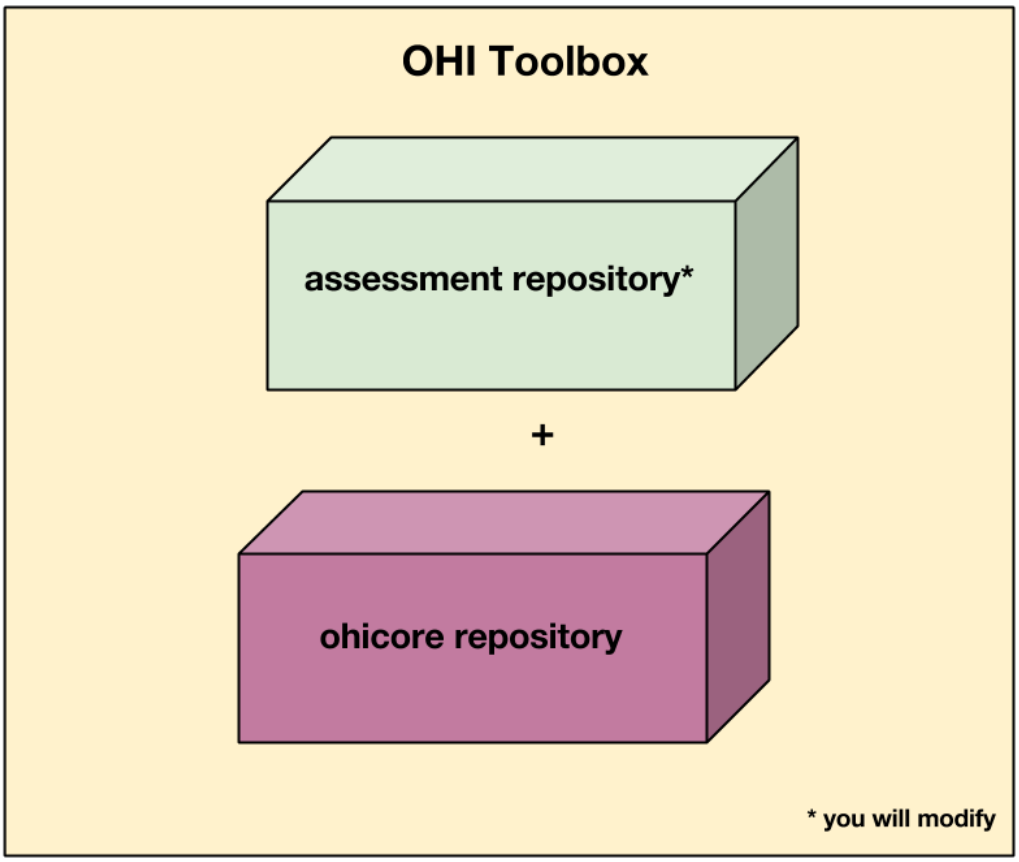



Figure 9: Toolbox = your repo + ohicore

Your **assessment repository** contains data input layers, configuration files, and scripts. These files are organized in the same way within a *scenario folder* called `subcountry2014`, with data layers, goal model equations, and configuration files from the global 2014 assessment. Files within the scenario folder are comma-separated-value (*.csv*) files and scripts written in the programming language *R*.

**Each OHI+ assessment repository has inputs and goal models based on the 2014 global assessment**. This means that each assessment repository isolates the information used for each region in the global assessment and stores it in a separate OHI+ assessment repository. Therefore, it can be an easy way to explore the inputs used in the global assessment for one specific place. When conducting an assessment, you will replace and modify as much of this information with local information that better represents your study area, since information reported at a national scale cannot always be attributed to its subcountry regions, as has been done in most cases in each OHI+ assessment repository. See more details in the discussion of the **layers folder**.

The Toolbox is open-source and can be downloaded and installed for free. You are able to navigate through these files both at `www.github.com/OHI-Science` and on your own computer once you have cloned the repository to your computer. GitHub is an online platform used by the OHI that facilitates collaboration and archives past versions of all files with the author identified. It can be accessed remotely by all members of your team and enables team members to synchronize their work together. Because all versions are saved, you can return to previous work and also compare different points in history to track how changes you make affect the output scores. Instruction on how to access your assessment repository is in the `Installing the Toolbox` section.

The following sections will describe the files included in the Toolbox. You will then learn what is required for data layers for your assessment and how to change goal models.

## 5.1 File system organization

This section is an orientation to the files within your assessment repository. The file system organization is the same for all assessment repositories, and can be viewed at `github.com/OHI-Science` or on your computer. While reading this section it is helpful to explore a repository at the same time to become familiar with its contents and structure. The following uses the assessment repository for Ecuador (*ecu*) as an example, available at www.github.com/OHI-Science/ecu.

**Most of your time will be spent preparing input layers and developing goal models**. You will also register prepared layers to be used in the goal models. This all will be an iterative process, but generally speaking you will work goal-by-goal, preparing the layers first, registering them, and then developing the goal models in *R*. to calculate the scores.

### 5.1.1 Assessment repositories

**Assessment repositories** are identified by a three-letter code; Ecuador's assessment repository is called '*ecu*'. Assessment repositories contain several things:

- The **scenario folder** is the most important folder within the repository; by default it is named `subcountry2014`. It contains all of the inputs needed to calculate OHI scores, and you will modify these inputs when conducting your assessment. The scenario folder is explained in detail in this section.
- All other files in the assessment repository are accessory files. Files with names beginning with a '.' are required for versioning capabilities by GitHub and do not appear when the assessment repository is viewed on your computer.

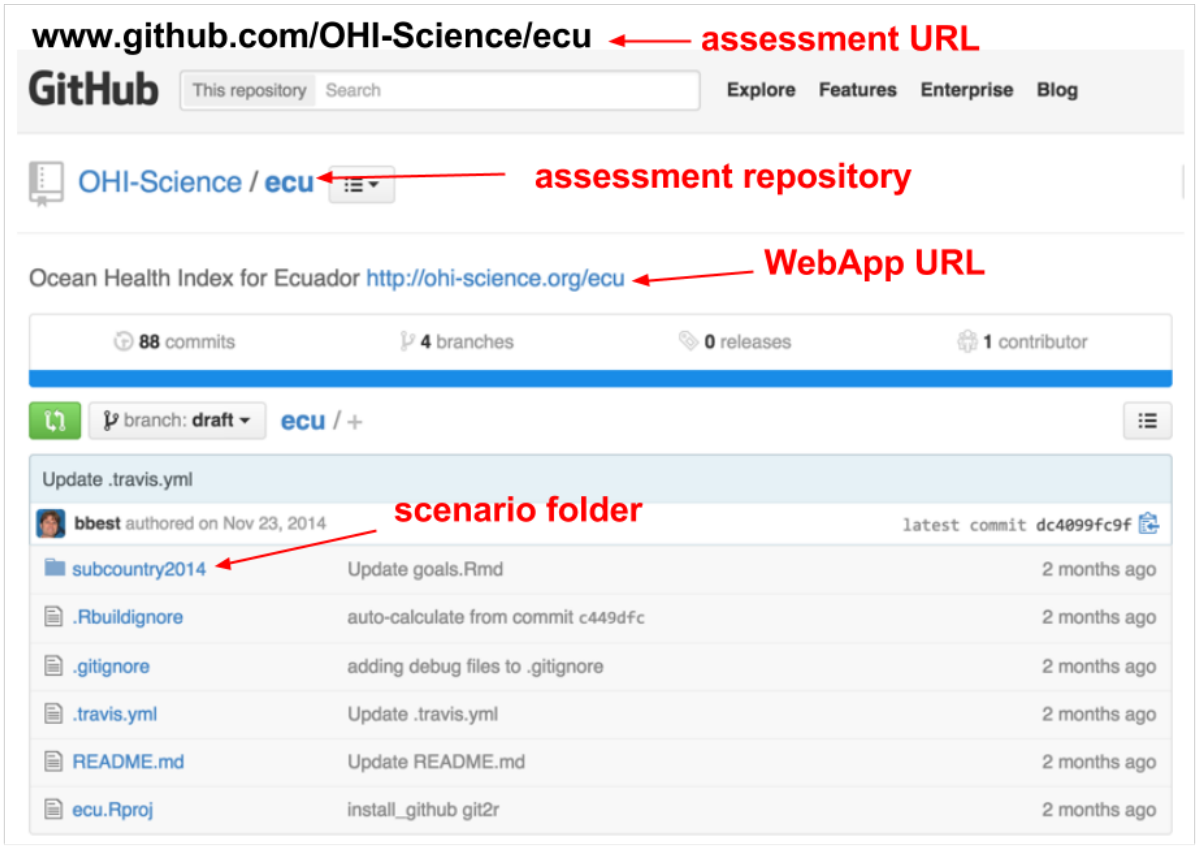

www.github.com/OHI-Science/ecu
assessment URL
GitHub
This repository
Search
Explore
Features
Enterprise
Blog
OHI-Science / ecu
assessment repository
Ocean Health Index for Ecuador http://ohi-science.org/ecu
WebApp URL
88 commits
4 branches
0 releases
1 contributor
branch: draft
ecu / +
Update .travis.yml
scenario folder
bbest authored on Nov 23, 2014
latest commit dc4099fc9f
subcountry2014
Update goals.Rmd
2 months ago
.Rbuildignore
auto-calculate from commit c449dfc
2 months ago
.gitignore
adding debug files to .gitignore
2 months ago
.travis.yml
Update .travis.yml
2 months ago
README.md
Update README.md
2 months ago
ecu.Rproj
install_github git2r
2 months ago

Figure 10:

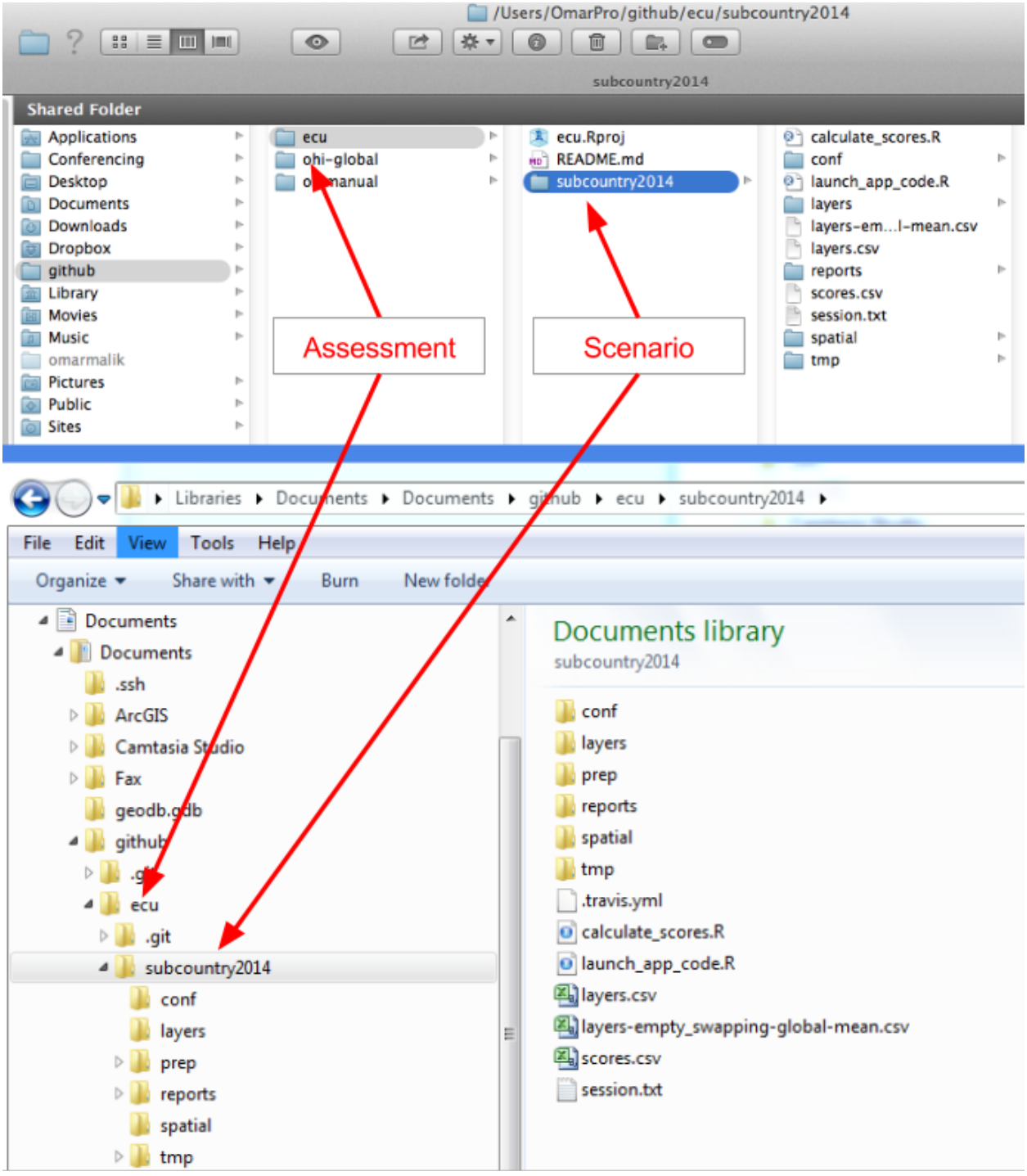



Figure 11: Navigating the assessment repository. The figure shows Mac folder navigation above and Windows navigation below.

### 5.1.2 Scenario folders

Scenario folders contain all files and scripts necessary to calculate OHI scores. There are two file types:

- ** *.csv* files** contain data inputs or configuration information.
- ** *.R* scripts** are written in the programming language R and use data inputs for processing and calculations.

There is one scenario folder in your assessment repository and it is called `subcountry2014` to indicate that the assessment is conducted at the subcountry scale (province, state, district, etc.), based on input layers and goal models used in the 2014 global assessment. When conducting your assessment, you can rename your scenario folder to reflect the subcountry regions in your study area and year the assessment was completed. For example, `province2015` would indicate the assessment was conducted for coastal provinces in the year 2015.

Once you complete your assessment with the `subcountry2014` (or equivalent) scenario, further assessments can be done simply by copying the `subcountry2014` folder and renaming it. This can be done for future assessments, for example `subcountry2016` or `subcountry2018`, which eventually would enable you to track changes in ocean health over time. You can also copy scenario folders to explore different policy and management scenarios, for example `subcountry2014_policy1`.

### 5.1.3 `subcountry2014` contents

This figure illustrates the files contained within the assessment repository's `subcountry2014` scenario folder. Important files are either *.csv* text files or *.R* script files. Files are organized into different folders within the `subcountry2014` folder, and you will modify some of these files while leaving others as they are.

Files and folders are presented here in alphabetical order. See the **Using the Toolbox** section for the workflow of how you will use the files.

**5.1.3.1 *calculate_scores.R*** `calculate_scores.R` is a script that runs everything required to calculate OHI scores using the prepared layers the `layers` folder that are registered in `layers.csv`. Scores will be saved in `scores.csv`.

**5.1.3.2 *conf* folder** The `conf` folder includes important configuration files required to calculate OHI scores. There are both *.R* scripts (`config.R` and `functions.R`) and *.csv* files (`goals.csv`, `pressures_matrix.csv`, `resilience_matrix.csv`, and `resilience_weights.csv`).

**5.1.3.3 *config.R*** `config.R` is an R script that configures labeling and constants appropriately. You will only need to modify this file when working with goals that have categories (example: habitat types or economy sectors) that are affected differently by pressures and resilience measures.

**5.1.3.4 *install_ohicore.R*** `install_ohicore.R` is a script that will install `ohicore`, which is the second repository required for the Toolbox and is the engine behind all OHI calculations. You will need to run this script only once when you begin.

**5.1.3.5 *functions.R*** `functions.R` is an R script containing the equations for each goal and sub-goal model. Each goal and sub-goal equation is stored as a separate function within the script. These functions calculate the status and trend using prepared layers saved in the 'layers' folder and registered in `layers.csv`. You will need to code in R to modify or develop new models, and it is best to work on one goal at a time.

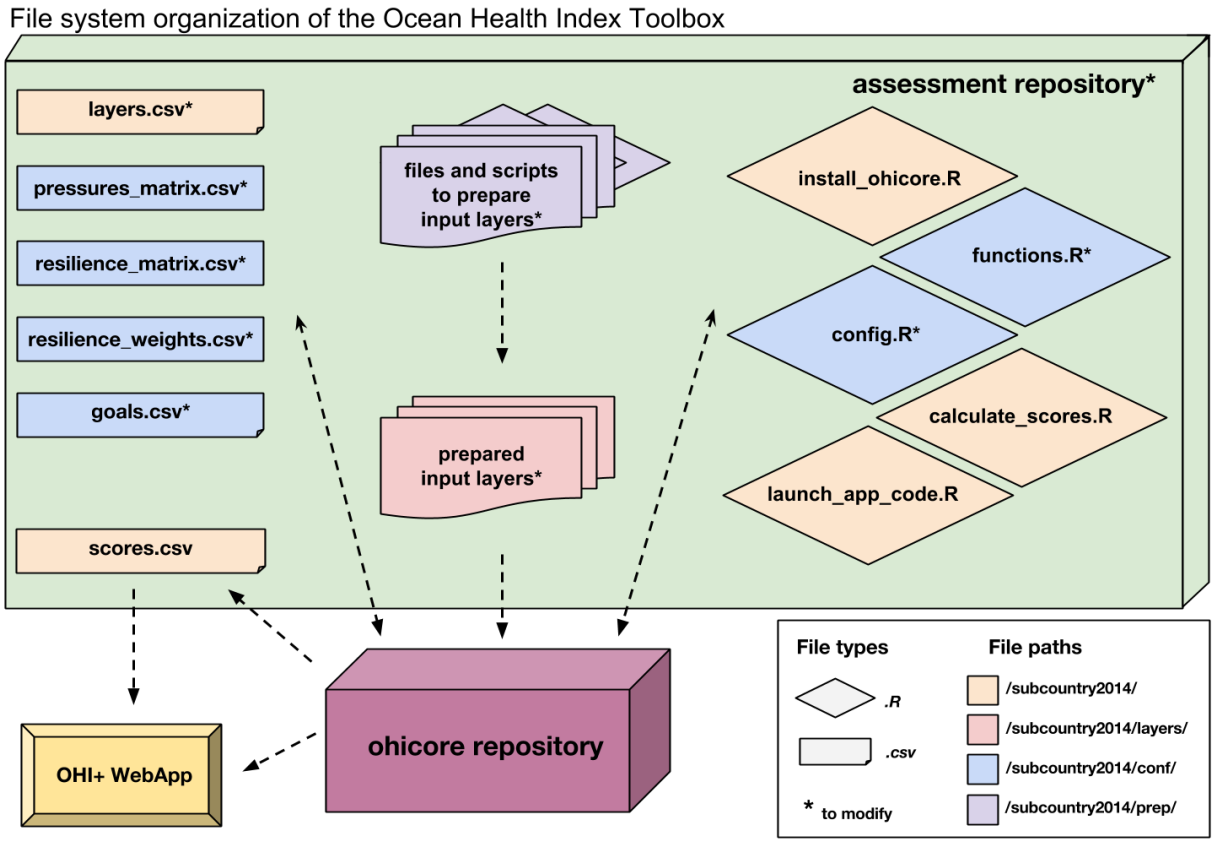



Figure 12: File system organization of the Ocean Health Index Toolbox

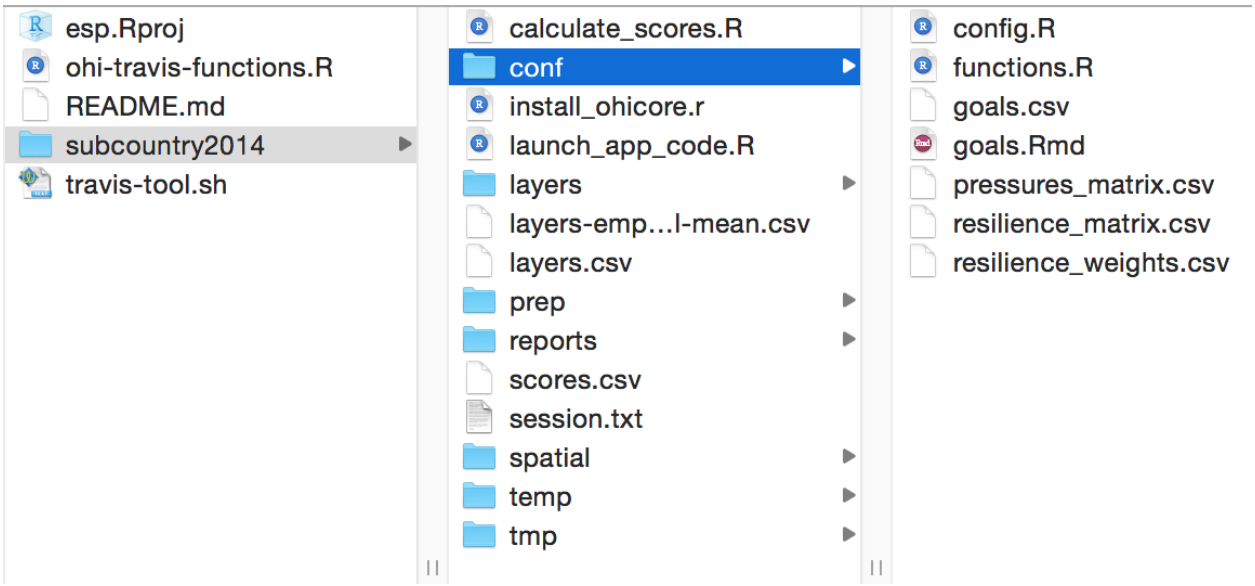



Figure 13: The `conf` folder contains important R functions and *.csv* files.

**5.1.3.6** ***goals.csv*** `goals.csv` is a table with information about goals and sub-goals. This includes the weight of each goal that is used to calculate the final Index scores when all goals are combined. Other information includes the goal description that is also presented in the WebApp. `goals.csv` also indicates the arguments passed to `functions.R`. These are indicated by two columns: *preindex_function* (functions for all goals that do not have sub-goals, and functions for all sub-goals) and *postindex_function* (functions for goals with sub-goals).

> TIP: It's important to check `goals.csv`'s weightings and preindex functions when you change goal or sub-goal model equations in `functions.r`.

**5.1.3.7** ***launch_app_code.R*** `launch_app_code.R` will launch a version of the App on your computer so that you can visualize any edits you make before synching to *github.com*.

**5.1.3.8** ***layers* folder** The `layers` folder contains all layers required to calculate goal scores, and each layer is an individual *.csv* file. The names of the *.csv* files within the layers folder correspond to those listed in the *filename* column of the `layers.csv`. All *.csv* files can be read in R, or with text editors or spreadsheet editors like Microsoft Excel.

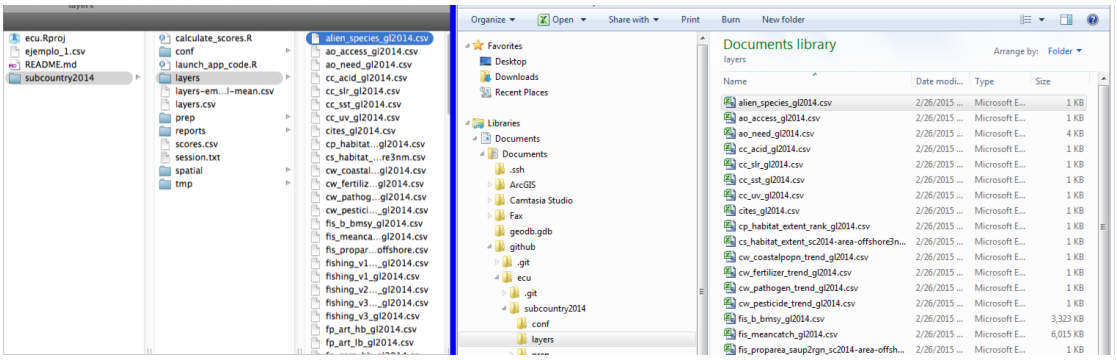



Figure 14: The `layers` folder contains every data layer as an individual *.csv* file. Mac navigation is shown on the left and Windows navigation is shown on the right.

Note that each *.csv* file within the `layers` folder has been formatted consistently. The Toolbox expects all data layers to be in the correct 'long format' and in separate files (see **Using the Toolbox**). Each file also has a column with unique region identifier (*rgn_id*). These numeric region identifiers have region names associated with them, that are set in `rgn_labels.csv` and can be modified.

> TIP: You can check your region identifiers (*rgn_id*) in the `rgn_labels.csv` file in the `layers` folder.

**5.1.3.8.1** ***_gl2014* and *_sc2014* suffixes** In your repository, layers are provided for your country based on input information from the 2014 global assessment. The global assessment had information for your country at the the spatial scale of the entire country, whereas your assessment has information for each subcountry region within your country. In most cases, layers from the global assessment was allocated equally to all regions in your study area (country). When this occurred, the layer was given a suffix of `_gl2014` to indicate that information is equal across all regions in the study area. While these layers may not provide much useful information to your assessment, the proper input structure is provided in these layers. Some layers contain information such as km2 of habitat that could not be equally allocated across all regions since this would provide a sum much greater than reality. In these cases, layers were down-weighted based on the proportion of offshore area or coastal population density. These layers have the suffix `_sc2014` with an indication of what was used to downweight. While this method removes any error of inflated sums and

provides the Toolbox with functioning layers, the allocation of these values may not be sensical to your study (i.e. if corals are only present in some regions of your study area but they are allocated to all).

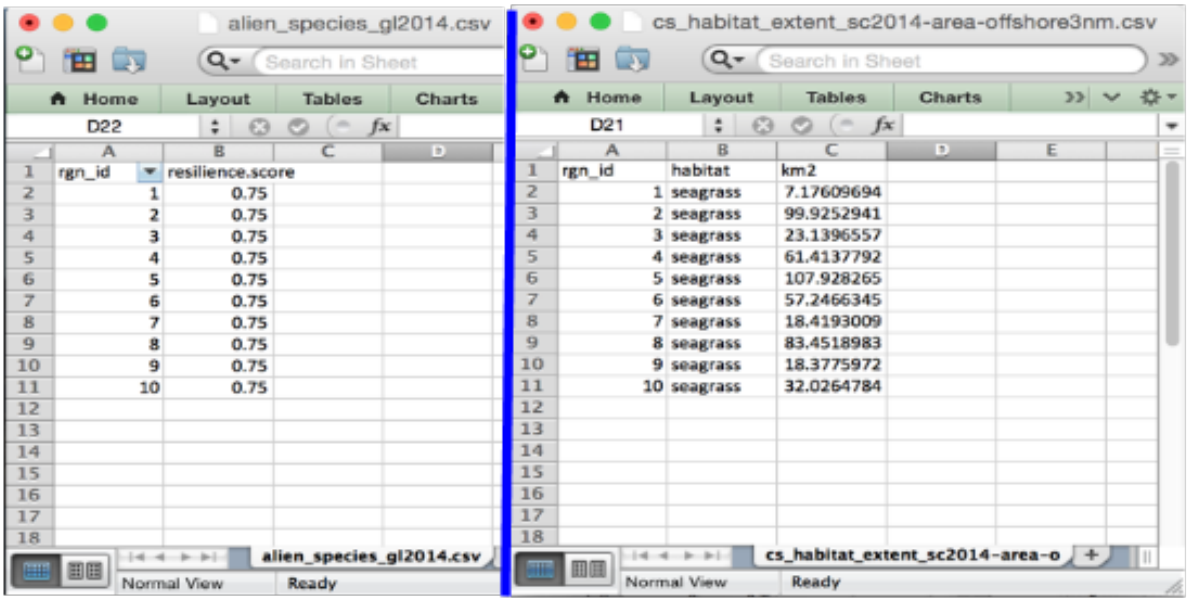



Figure 15: Differences in input layers with equal information for each region (suffixed with `_gl2014`) and weighted information for each region (suffixed with `_sc2014`).

**5.1.3.9** ***layers-empty_swapping-global-mean.csv*** `layers-empty_swapping-global-mean.csv` contains a list of layers where information for your country was not available for the global assessment. For the Toolbox to be able to run, these layers were filled with averages from all other countries included in the global assessment. This file is not used anywhere by the Toolbox but is a registry of layers that should prioritized to be replaced with your own local layers if you require these layers for the models you develop.

**5.1.3.10** ***layers.csv*** The `layers.csv` file is the registry and directory that manages all data required for you assessment. All relevant input information is prepared as individual data layers and then registered in this file. The Toolbox uses `layers.csv` to access the proper input information and display information on the WebApp. You will update some of the columns in `layers.csv`, and some of them will be auto-generated by the Toolbox code when it is running.

> TIP: `layers.csv` is a very useful reference throughout the assessment process.

`layers.csv` is easiest to view in spreadsheet software (i.e. Microsoft Excel). When you open it, you will see that each row of information represents an individual input layer that has been prepared for the Toolbox. The first columns contain information that will be updated by your team as you incorporate modified or new layers; all other columns are generated later by the Toolbox as it confirms data formatting and content and alerts you of any formatting inconsistencies. The columns you will update are: *targets, layer, name, description, fld_value, units, filename.*

**Columns you will update**

- **targets** indicates which goal or dimension uses the layer. Goals are indicated with two-letter codes and sub-goals are indicated with three-letter codes (see the table just below). Pressures, resilience, and spatial layers indicated separately.

| | targets | layer | name | description | fld_value | units | filename |
|---|---|---|---|---|---|---|---|
| 1 | | | | | | | |
| 2 | AO | ao_access | Fisheries manage | The opportunity fo | value | value | ao_access.csv |
| 3 | AO | ao_need | Purchasing powe | The per capita pu | value | value | ao_need.csv |
| 4 | CW | cw_coastalpopn_tren | Coastal human p | Coastal populatio | trend | trend score | cw_coastalpopn_tr |
| 5 | CW | cw_fertilizer_trend | Fertilizer consum | Statistics on fertili | trend.score | trend score | cw_fertilizer_trend. |
| 6 | CW | cw_pathogen_trend | Trends in access | Trends in percent | trend | trend score | cw_pathogen_trend |
| 7 | CW | cw_pesticide_trend | Pesticide consum | Statistics on pesti | trend.score | trend score | cw_pesticide_trend |
| 8 | FIS | fis_b_bmsy | B/Bmsy estimates | obtained using th | b_bmsy | B / B_msy | fis_b_bmsy.csv |
| 9 | FIS | fis_meancatch | Catch data for ea | Reported data inc | mean_catch | metric tons | fis_meancatch.csv |

Figure 16:

- **layer** is the identifying name of the input layer that will be used in R scripts like `functions.R` and *.csv* files like `pressures_matrix.csv` and `resilience_matrix.csv`. This is also displayed on the WebApp under the drop-down menu when the variable type is 'input layer'.
- **name** is a longer title of the input layer; this is displayed on the WebApp under the drop-down menu when the variable type is 'input layer'.
- **description** is further description of the input layer, including the source of the original data. This is also displayed on the WebApp under the drop-down menu when the variable type is 'input layer'.
- **fld_value** the values' units in the input layer. The information in this column must match the column header in the input layer.
- **units** the values' units in the input layer. This differs from *fld_value* above as the *units* column is displayed on the WebApp and can have more descriptive naming.
- **filename** is the input layer itself. This file has input information for each region within the study area, and is located in the `subcountry2014/layers` folder.

| Goal | Subgoal | 2- or 3- letter code |
|---|---|---|
| Food Provision | | FP |
| | Fisheries | FIS |
| | Mariculture | MAR |
| Artisanal Fishing Opportunity | | AO |
| Natural Products | | NP |
| Coastal Protection | | CP |
| Carbon Storage | | CS |
| Livelihoods and Economies | | LE |
| | Livelihoods | LIV |
| | Economies | ECO |
| Tourism and Recreation | | TR |
| Sense of Place | | SP |
| | Lasting Special Places | LSP |
| | Iconic Species | ICO |
| Clean Waters | | CW |
| Biodiversity | | BD |
| | Habitats | HAB |
| | Species | SPP |

**5.1.3.11** ***prep* folder** The `prep` folder is included in your repository so that layer preparation can be collaborative and version controlled. It is not necessary to use this folder but you may find it useful as other assessments have.

**5.1.3.12** ***pressures_matrix.csv*** `pressures_matrix.csv` is a table that indicates which individual pressures (stressors) affect which goal, sub-goals, or components, and weights them from 1-3 (a weight of 0 is shown as a blank). These weights are relative to each row of the matrix (goal, sub-goal, or component). These weights are used in global assessments based on scientific literature and expert opinion, and you can modify the weightings if necessary for your assessment. The pressures matrix is the same as Table S25 in the Supplementary Information for Halpern *et al.* 2012.

Each pressure (column) of the pressures matrix is the layer name of the pressures layer file that is saved in the `layers` folder and is registered in `layers.csv`. Pressures layers have values for every region in the study

**5.1.3.13** ***reports* folder** The `reports` folder contains flower plots and tables for every region in the study area and for the study area itself, which by convention is called 'GLOBAL' in these files.

**5.1.3.14** ***resilience_matrix.csv*** `resilience_matrix.csv` is a table that indicates which individual resilience measures affect which goal, sub-goals, or components. Like the pressures matrix, the resilience matrix also has weights, but these weights depend on the level of information available. These weights are stored in a separate file in the `conf` folder: `resilience_weights.csv`. The resilience matrix is the same as Table S26 in the Supplementary Information for Halpern *et al.* 2012.

Each resilience measure (column) of the resilience matrix is the layer name of the resilience layer file that is saved in the `layers` folder and is registered in `layers.csv`. Resilience layers have values for every region in the study area. Resilience values are scaled such that all values range from 0-1.

**5.1.3.15** ***resilience_weights.csv*** `resilience_weights.csv` is a table that indicates the weight of each resilience layer based on the level of information available.

**5.1.3.16** ***scores.csv*** `scores.csv` is a text file containing the calculated scores for each dimension (future, pressures, resilience, score, status, trend) for each region in the study area. Regions have the numeric identifiers set in `subcountry2014/layers/rgn_labels.csv` and the study area has the numeric identifier of 0. Scores are calculated with registered layers in `layers.csv`: when you begin an assessment this will be information for your country from the global 2014 assessment and goal models from the global 2014 assessment. Scores from `scores.csv` are viewed through the WebApp using the 'Output Score' pulldown menu on the 'App' page.

**5.1.3.17** ***session.txt*** `session.txt` is not used in OHI calculations but stores information about how the Toolbox was installed which may be useful for debugging purposes.

**5.1.3.18** ***spatial* folder** The `spatial` folder contains two spatial files: `regions_gcs.geojson` and `regions_gcs.js`. These files spatially identifies the study area and regions for the assessment and are stored in the JSON and GeoJSON formats that can be displayed by the App. If you plan to redefine the spatial boundaries for your assessment, you will need to provide a shapefile to the OHI+ development team and we will create the proper `regions_gcs.geojson` and `regions_gcs.js` files for you. You will need a spatial analyst to do this: see the **Defining spatial boundaries** section for instruction.

**5.1.3.19** ***temp* or *tmp* folders** Contents within the `temp` or `tmp` folders are not used to calculate scores but can be used for temporary organization for your assessment.

## 5.2 Formatting Data for the Toolbox

### 5.2.1 Introduction

The OHI Toolbox is designed to work in the programming language **R** using input data stored in text-based *.csv* files (*csv* stands for 'comma-separated value'; these files can be opened as a spreadsheet using Microsoft Excel or similar programs). Each data layer (data input) has its own *.csv* file, which is combined with others within the Toolbox for the model calculations. These data layers are used for calculating goal scores, meaning that they are inputs for status, trend, pressures, and resilience. The global analysis included over 100 data layer files, and there will probably be as many in your own assessments. This section describes and provides examples of how to format the data layers for the Toolbox.

OHI goal scores are calculated at the scale of the reporting unit, which is called a '**region**' and then combined using an area-weighted average to produce the score for the overall area assessed, called a '**study area**'. The

OHI Toolbox expects each data file to be in a specific format, with data available for every region within the study area, with data layers organized in 'long' format (as few columns as possible), and with a unique region identifier (*rgn_id*) associated with a single *score* or *value*. In order to calculate trend, input data must be available as a time series for at least 5 recent years (and the longer the time series the better, as this can be used in setting temporal reference points).

The example below shows information for a study area with 4 regions. There are two different (and separate) data layer files: tourism count (`tr_total.csv`) and natural products harvested, in metric tonnes (`np_harvest_tonnes.csv`). Each file has data for four regions (1-4) in different years, and the second has an additional 'categories' column for the different types of natural products that were harvested. In this example, the two data layers are appropriate for status calculations with the Toolbox because:

1. At least five years of data are available,
2. There are no data gaps
3. Data are presented in 'long' or 'narrow' format (not 'wide' format – see "**Long Formatting**"" section).

**Example of data in the appropriate format:**

| rgn_id | year | count |
|---|---|---|
| 1 | 2005 | 177.14 |
| 1 | 2006 | 201.39 |
| 1 | 2007 | 199.81 |
| 1 | 2008 | 212.99 |
| 1 | 2009 | 228.81 |
| 2 | 2005 | 580.98 |
| 2 | 2006 | 730.18 |
| 2 | 2007 | 717.00 |
| 2 | 2008 | 851.44 |
| 2 | 2009 | 836.68 |
| 3 | 2005 | 129.69 |
| 3 | 2006 | 173.45 |
| 3 | 2007 | 229.86 |
| 3 | 2008 | 231.44 |
| 3 | 2009 | 221.95 |
| 4 | 2005 | 397.00 |
| 4 | 2006 | 566.00 |
| 4 | 2007 | 591.00 |
| 4 | 2008 | 1154.00 |
| 4 | 2009 | 1570.00 |

| rgn_id | product | year | tonnes |
|---|---|---|---|
| 1 | ornamentals | 2005 | 10327 |
| 1 | ornamentals | 2006 | 10389 |
| 1 | ornamentals | 2007 | 10897 |
| 1 | ornamentals | 2008 | 9985 |
| 1 | ornamentals | 2009 | 9001 |
| 2 | shells | 2005 | 6179 |
| 2 | shells | 2006 | 6823 |
| 2 | shells | 2007 | 8239 |
| 2 | shells | 2008 | 8819 |
| 2 | shells | 2009 | 9205 |
| 3 | coral | 2005 | 22079 |
| 3 | coral | 2006 | 25297 |
| 3 | coral | 2007 | 25361 |
| 3 | coral | 2008 | 23817 |
| 3 | coral | 2009 | 23623 |
| 4 | shells | 2005 | 7500 |
| 4 | shells | 2006 | 9700 |
| 4 | shells | 2007 | 8600 |
| 4 | shells | 2008 | 9400 |
| 4 | shells | 2009 | 9300 |

Figure 17:

### 5.2.2 Gapfilling

It is important that data prepared for the Toolbox have no missing values or 'gaps'. Data gaps can occur in two main ways: 1) **temporal gaps**: when several years in a time series in a single region have missing data, and 2) **spatial gaps**: when all years for a region have missing data (and therefore the whole region is 'missing' for that data layer).

How these gaps are filled will depend on the data and regions themselves, and requires thoughtful, logical decisions to most reasonably fill gaps. Each data layer can be gapfilled using different approaches. Some data layers will require both temporal and spatial gapfilling. The examples below highlight some example of temporal and spatial gapfilling.

All decisions of gapfilling should be documented to ensure transparency and reproducibility. The examples below are in Excel, but programming these changes in software like R is preferred because it promotes easy transparency and reproducibility.

**5.2.2.1 Temporal gapfilling** Temporal gaps occur when a region is missing data for some years. The Toolbox requires data for each year for every region. It is important to make an informed decision about how to temporally gapfill data.

| rgn_id | year | count |
|---|---|---|
| 1 | 2005 | 177.14 |
| 1 | 2006 | |
| 1 | 2007 | |
| 1 | 2008 | 212.99 |
| 1 | 2009 | 228.81 |
| 2 | 2005 | 580.98 |
| 2 | 2006 | 730.18 |
| 2 | 2007 | 717.00 |
| 2 | 2008 | 851.44 |
| 2 | 2009 | 836.68 |
| 3 | 2005 | 129.69 |
| 3 | 2006 | 173.45 |
| 3 | 2007 | 229.86 |
| 3 | 2008 | 231.44 |
| 3 | 2009 | |
| 4 | 2005 | 397.00 |
| 4 | 2006 | 566.00 |
| 4 | 2007 | |
| 4 | 2008 | 1154.00 |
| 4 | 2009 | 1570.00 |

In this data layer, there are three regions that have missing values for one or more years and will require temporal gapfilling: Regions 1, 3, 4.

No regions in this example require spatial gapfilling.

Figure 18:

Often, regression models are the best way to estimate data and fill temporal gaps. Here we give an example that assumes a linear relationship between the year and value variables within a region. If data do not fit a linear framework, other models may be fit to help with gapfilling. Here we give an example assuming linearity.

Using a linear model can be done in most programming languages using specific functions, but here we show this step-by-step using functions in Excel for Region 1.

**Temporal gapfilling example (assumes linearity: able to be represented by a straight line on a graph)):**

There are four steps to temporally gapfill with a linear model, illustrated in the figures with four columns.

**1. Calculate the slope for each region**

The first step is to calculate the slope of the line that is fitted through the available data points. This can be done in Excel using the **SLOPE(known_y's,known_x's)** function as highlighted in the figure below. In this case, the x-axis is *years* (2005, 2006, etc. . . ), the y-axis is *count*, and the Excel function automatically plots and fits a line through the known values (177.14 in 2005, 212.99 in 2008, and 228.81 in 2009), and subsequently calculates the slope (12.69).

| | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|
| 39 | | | | | | Steps to temporally gapfill data: | | | |
| 40 | | | | | | 1 | 2 | 3 | 4 |
| 41 | rgn_id | year | count | | | Slope | Intercept | y = mx + b | Value (final) |
| 42 | 1 | 2005 | 177.14 | | | | | 176.70 | 177.14 |
| 43 | 1 | 2006 | | | | | | 189.39 | 189.39 |
| 44 | 1 | 2007 | | | | | | 202.08 | 202.08 |
| 45 | 1 | 2008 | 212.99 | | | | | 214.78 | 212.99 |
| 46 | 1 | 2009 | 228.81 | | | 12.69 | -25273.89 | 227.47 | 228.81 |

Formula: SLOPE( $D$42:$D$46 , $C$42:$C$46 )

Figure 19:

**2. Calculate the y-intercept for each region**

The next step is to calculate the intercept of the line that is fitted through the available data points. This can be done in Excel similarly as for the slope calculation, using the the **INTERCEPT(known_y's,known_x's)** function that calculates the y-intercept (-25273.89) of the fitted line.

**3. Calculate y for all years**

The slope and y-intercept that were calculated in steps 1 and 2 can then be used along with the year (independent variable) to calculate the unknown 'y-values'. To do so, simply replace the known three values

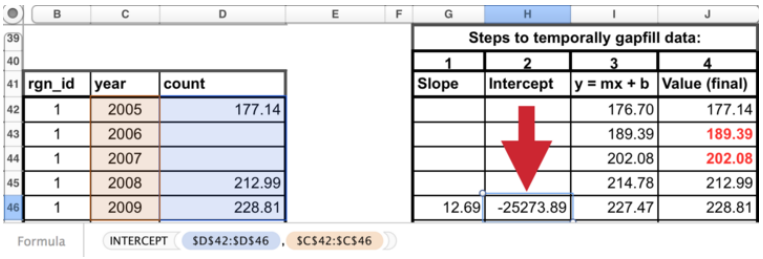



Figure 20:

into the $\mathbf{y = mx + b}$ equation (m=slope, x=year, b=intercept), to calculate the unknown 'count' for a given year (189.39 in 2006, and 202.08 in 2007).

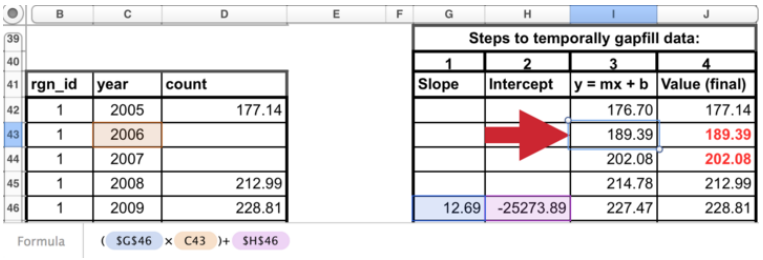



Figure 21:

**4. Replace modeled values into original data where gaps had occurred**

Substitute these modeled values that were previously gaps in the timeseriew. *The data layer is now ready for the Toolbox, gapfilled and in the appropriate format.*

**5.2.2.2 Spatial gapfilling** Spatial gaps are when no data are available for a particular region. The Toolbox requires data for each region. It is important to make an informed decision about how to spatially gapfilling data.

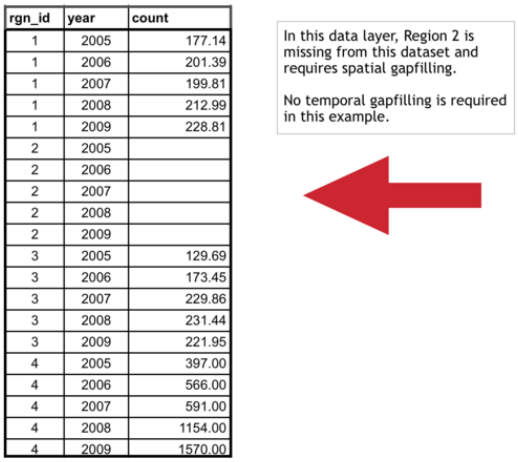



Figure 22:

To fill gaps spatially, you must assume that one region is like another, and data from another region is adequate to be substituted in place of the missing data. This will depend on the type of data and the properties of the

regions requiring gapfilling. For example, if a region is missing data but has similar properties to a different region that does have data, the missing data could be 'borrowed' from the region with information. Each data layer can be gapfilled using a different approach when necessary.

**Characteristics of regions requiring gapfilling that can help determine which type of spatial gapfilling to use:**

1. proximity: can it be assumed that nearby regions have similar properties?
2. study area: are data reported for the study area, and can those data be used for subcountry regions?
3. demographic information: can it be assumed a region with a similar population size has similar data?

**Spatial gapfilling example:**

For a certain data layer, suppose the second region (*rgn_id 2*) has no data reported, as illustrated in the figure above. How to spatially gapfill *rgn_id 2* requires thinking about the properties and characteristics of the region and the data, in this case, tourist count.

Here are properties that can be important for decision making:

*rgn_id 2*:

- is located between *rgn_id 1* and 3
- is larger than *rgn_id 1*
- has similar population size/demographics to *rgn_id 3*
- has not been growing as quickly as *rgn_id 4*

There is no absolute answer of how to best gapfill *rgn_id 2*. Here are a few reasonable possibilities:

Assign *rgn_id 2* values from:

- *rgn_id 1* because it is in close proximity to *rgn_id 2*
- *rgn_id 3* because it is in close proximity to *rgn_id 2* and has similar population size/demographics
- *rgn_id 1* and 3 averaged since they are in close proximity to *rgn_id 2*

Suppose the decision was made to gapfill *rgn_id 2* using the mean of *rgn_id 1* and *3* since this would use a combination of both of those regions. Again, other possibilities could be equally correct. But some form of spatial gapfilling is required so a decision must be made. The image below illustrates this in Excel.

The data layer is now ready for the Toolbox, gapfilled and in the appropriate format.

### 5.2.3 Long formatting

The Toolbox expects data to be in 'long' or 'narrow' format. Below are examples of correct and incorrect formatting, and tips on how to transform data into the appropriate format.

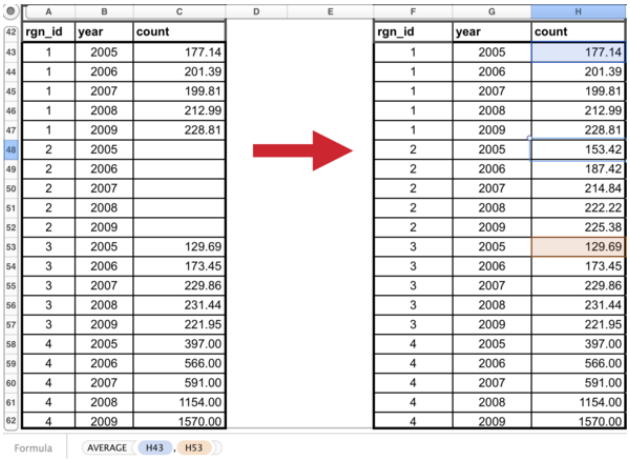
**Example of data in an incorrect format:**

With 'wide' format, data layers are more difficult to combine with others and more difficult to read and to analyze.

**Transforming data into 'narrow' format:**

Data are easily transformed in a programming language such as R.

In R, the `reshape` package has the `melt` command, which will melt the data from a wide format into a narrow format. It also can `cast` the data back into a wide format if desired. R documentation:



| rgn_id | year | count |
|---|---|---|
| 1 | 2005 | 177.14 |
| 1 | 2006 | 201.39 |
| 1 | 2007 | 199.81 |
| 1 | 2008 | 212.99 |
| 1 | 2009 | 228.81 |
| 2 | 2005 | |
| 2 | 2006 | |
| 2 | 2007 | |
| 2 | 2008 | |
| 2 | 2009 | |
| 3 | 2005 | 129.69 |
| 3 | 2006 | 173.45 |
| 3 | 2007 | 229.86 |
| 3 | 2008 | 231.44 |
| 3 | 2009 | 221.95 |
| 4 | 2005 | 397.00 |
| 4 | 2006 | 566.00 |
| 4 | 2007 | 591.00 |
| 4 | 2008 | 1154.00 |
| 4 | 2009 | 1570.00 |

| rgn_id | year | count |
|---|---|---|
| 1 | 2005 | 177.14 |
| 1 | 2006 | 201.39 |
| 1 | 2007 | 199.81 |
| 1 | 2008 | 212.99 |
| 1 | 2009 | 228.81 |
| 2 | 2005 | 153.42 |
| 2 | 2006 | 187.42 |
| 2 | 2007 | 214.84 |
| 2 | 2008 | 222.22 |
| 2 | 2009 | 225.38 |
| 3 | 2005 | 129.69 |
| 3 | 2006 | 173.45 |
| 3 | 2007 | 229.86 |
| 3 | 2008 | 231.44 |
| 3 | 2009 | 221.95 |
| 4 | 2005 | 397.00 |
| 4 | 2006 | 566.00 |
| 4 | 2007 | 591.00 |
| 4 | 2008 | 1154.00 |
| 4 | 2009 | 1570.00 |

Figure 23:

| | | Year | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Region | DataLayer | 2000 | 2001 | 2002 | 2003 | 2004 | 2005 | 2006 | 2007 | 2008 | 2009 |
| A | GDP_USDx1000 | 8 | 7 | 30 | 26 | 69 | 39 | 108 | 92 | 261 | 151 |
| B | GDP_USDx1000 | 13 | 9 | 13 | 14 | 10 | 12 | 14 | 10 | 6 | 5 |
| C | GDP_USDx1000 | 2132 | 2325 | 2963 | 3214 | 2942 | 2910 | 1759 | 2029 | 2077 | 2453 |
| D | GDP_USDx1000 | 21 | 5 | 14 | 2 | 11 | 3 | 26 | 14 | 15 | 100 |
| A | governance_indicator | | | | | | 0.8545 | 0.5400 | 0.706 | 1 | 1 |
| B | governance_indicator | | | | | | 0.8564 | 0.7794 | 0.861 | 1 | 1 |
| C | governance_indicator | | | | | | 0.8779 | 1 | 1 | 0.898 | 1 |
| D | governance_indicator | | | | | | 0.8537 | 0.5373 | 0.704 | 1 | 1 |

Figure 24:

- http://cran.r-project.org/web/packages/reshape2/reshape2.pdf
- http://www.slideshare.net/jeffreybreen/reshaping-data-in-r
- http://tgmstat.wordpress.com/2013/10/31/reshape-and-aggregate-data-with-the-r-package-reshape2/

Example code using the *melt* command in the *reshape2* library. Assume the data above is in a variable called *data_wide*:

```
install.packages('reshape2')
library(reshape2)
data_melt = melt(data=data_wide, id.vars=c('Region', 'DataLayer'), variable.name='Year')
data_melt = data_melt[order(data_melt$DataLayer, data_melt$Region),]
```

Figure 25:

This will melt everything except any identified columns (*Region* and *DataLayer*), and put all other column headers into a new column named *Year*. Data values will then be found in a new column called *value*.

The final step is optional: ordering the data will make it easier for humans to read (R and the Toolbox can read these data without this final step):

**Example of data in the appropriate (long) format:**

# 6 Installing the Toolbox

**Section Summary:**

GDP_USDx1000.csv

| Region | Year | value |
|---|---|---|
| A | 2000 | 8 |
| A | 2001 | 7 |
| A | 2002 | 30 |
| A | 2003 | 26 |
| A | 2004 | 69 |
| A | 2005 | 39 |
| A | 2006 | 108 |
| A | 2007 | 92 |
| A | 2008 | 261 |
| A | 2009 | 151 |
| B | 2000 | 13 |
| B | 2001 | 9 |
| B | 2002 | 13 |
| B | 2003 | 14 |
| B | 2004 | 10 |
| B | 2005 | 12 |
| B | 2006 | 14 |
| B | 2007 | 10 |
| B | 2008 | 6 |
| B | 2009 | 5 |

governance_indicator.csv

| Region | Year | value |
|---|---|---|
| A | 2000 | |
| A | 2001 | |
| A | 2002 | |
| A | 2003 | |
| A | 2004 | |
| A | 2005 | 0.854599407 |
| A | 2006 | 0.540059347 |
| A | 2007 | 0.706231454 |
| A | 2008 | 1 |
| A | 2009 | 1 |
| B | 2000 | |
| B | 2001 | |
| B | 2002 | |
| B | 2003 | |
| B | 2004 | |
| B | 2005 | 0.856410256 |
| B | 2006 | 0.779487179 |
| B | 2007 | 0.861538462 |
| B | 2008 | 1 |
| B | 2009 | 1 |

Figure 26:

In this section, you will learn how to successfully download, install, and use the software required to conduct an assessment. You will create a GitHub account and install R, RStudio, git, and the Github desktop app. OHI assessments are conducted through open-source platforms that allow you to make real-time changes with collaborators, and to track progress so that errors can be corrected and new insights can be shared in the future.

## 6.1 Overview

Your assessment repository is located at *github.com/OHI-Science* and we recommend saving it to your computer so that you can sync changes back online to save versions and faciliate collaboration. Conducting an OHI assessment using GitHub enables collaboration and transparency, and will provide access to the latest developments in the Toolbox software, allowing the OHI team to provide support remotely if necessary.

This section explains the GitHub workflow and how to access and setup required software. You can use GitHub to upload any modifications you make so that you can work collaboratively with your team.

**Required software:**

1. **Github App**
2. ** *git* **
3. **R**
4. **RStudio**


Figure 27:

## 6.2 GitHub

**GitHub** is an open-source development platform that enables easy collaboration and versioning, which means that all saved versions are archived and attributed to each user. It is possible to revert back to any previous version, which is incredibly useful to not only to document what work has been done, but how it differs from work done in the past, and who is responsible for the changes.

**GitHub Vocabulary:**

- **clone** ~ download to your computer from online version with synching capabilities enabled
- **commit** ~ message associated with your changes at a point in time
- **pull** ~ sync a repo on your computer with online version
- **push** ~ sync the online repo with your version, only possible after committing

**sync = pull + commit + push**

### 6.2.1 Learning GitHub

The following section describes how to use GitHub to access and sync your assessment repository. There are also many great resources available online with more in-depth information:

- **Git and GitHub** by Hadley Wickham: http://r-pkgs.had.co.nz/git.html
- **Collaboration and Time Travel: Version Control with Git, GitHub and RStudio** video tutorial by Hadley Wickham: www.rstudio.com/resources/webinars
- **Good Resources for Learning Git and GitHub** by GitHub: https://help.github.com/articles/good-resources-for-learning-git-and-github/

## 6.3 Accessing GitHub Repositories

GitHub has an online interface and a desktop application for the version-control software called ** *git*. ** In addition to cloning your GitHub repository to your computer, you will need to download and install *git* software and the GitHub App (application), both of which are freely available.

### 6.3.1 Create a GitHub account

Create a GitHub account at http://github.com. Choose a username and password. You will use this username and password when you install and set up *git* on your computer.

### 6.3.2 Install *git* software

How you install *git* will depend on whether you are working on a Windows or Mac computer. It will also depend on your operating system version. If you have problems following these instructions, it is likely because your operating system requires a previous version of *git*. Previous versions are available from http://www.wandisco.com/git/download (you will need to provide your email address).

**For Windows:**

- Download *git* at http://git-scm.com/downloads and follow the install instructions.
- When running the Windows installer, use all default options except "Adjusting your PATH environment": instead, select "***Run Git from the Windows Command Prompt***". This will allow later compatibility with RStudio.

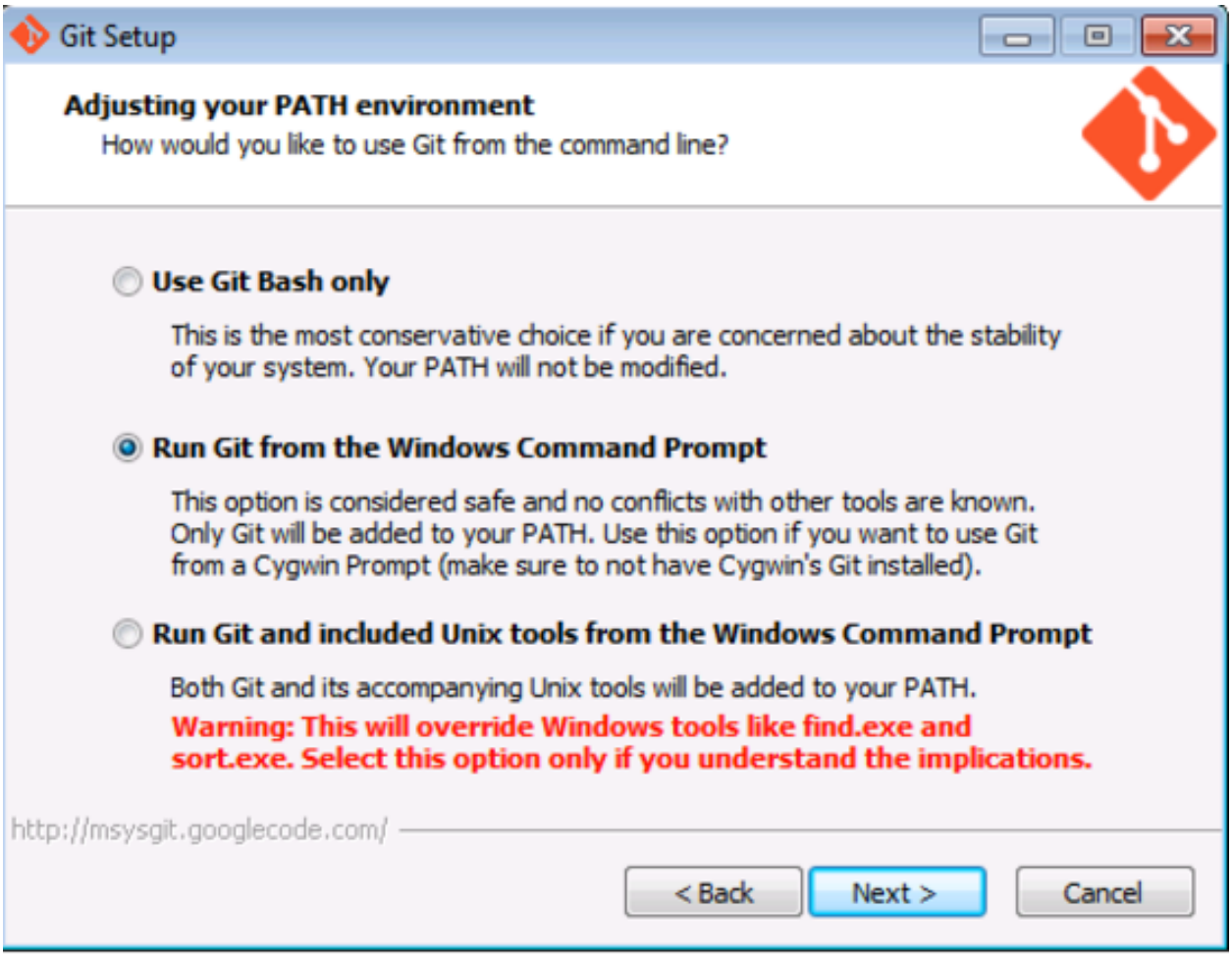

Git Setup
Adjusting your PATH environment
How would you like to use Git from the command line?
Use Git Bash only
This is the most conservative choice if you are concerned about the stability of your system. Your PATH will not be modified.
Run Git from the Windows Command Prompt
This option is considered safe and no conflicts with other tools are known. Only Git will be added to your PATH. Use this option if you want to use Git from a Cygwin Prompt (make sure to not have Cygwin's Git installed).
Run Git and included Unix tools from the Windows Command Prompt
Both Git and its accompanying Unix tools will be added to your PATH.
Warning: This will override Windows tools like find.exe and sort.exe. Select this option only if you understand the implications.
http://msysgit.googlecode.com/
< Back
Next >
Cancel

Figure 28:

**For Mac:**

- Download *git* at http://git-scm.com/downloads and follow the install instructions.
- Apple's Xcode has a command line tools option during install which can override the preferred *git* command line tools. To ensure you are using the latest preferred version of *git*, you will need to launch Terminal and type the following few lines of code:
- Access Terminal from the Applications folder: **Applications > Utilities > Terminal**. When you launch Terminal a window will appear with your computer's name followed by a `$`. When you type, your commands will appear after the `$`.

Add access your 'bash profile' by typing:

```
pico ~/.bash_profile
```

You are now able to edit your 'bash profile'. Type:

```
export PATH=/usr/local/git/bin:$PATH
```

Exit pico by typing:

control-X
y
return/enter

Exit Terminal by typing:

```
exit
```

Finally, quit Terminal.

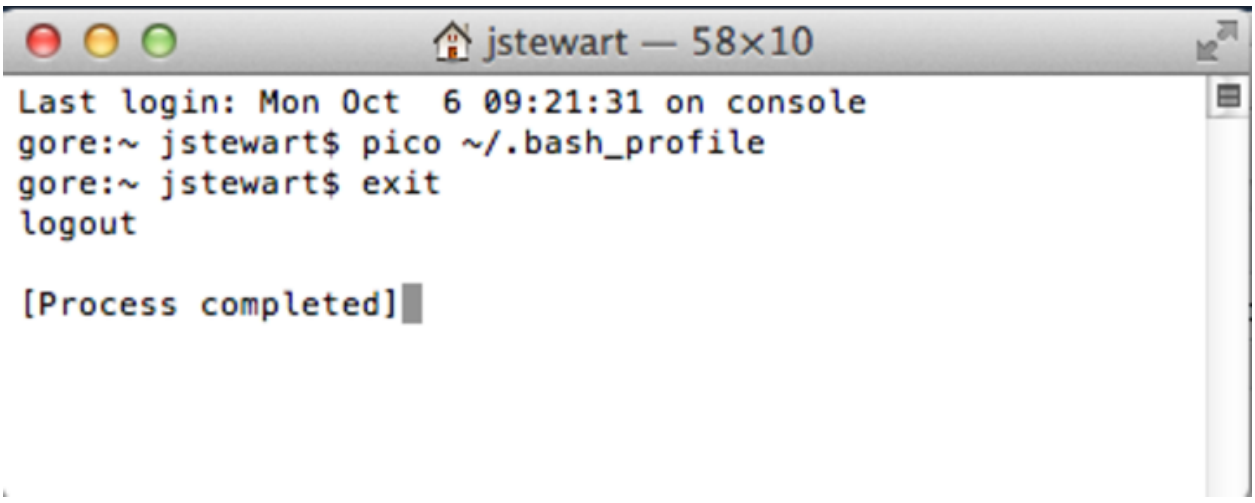



Figure 29:

### 6.3.3 Set up your Git Identity

After downloading and installing *git*, you will need to set up your **Git Identity**, which identifies you with your work. *Note*: if you have any problems with the following instructions, it is likely because of incompatibility between the version of your operating system and the version of git you downloaded in the previous section. In this case, find and download a compatible version at www.wandisco.com/git/download and then follow the instructions below.

You will set up your GitHub identity using the command line specific to Windows or Mac:

- **Windows**: Start > Run > cmd
- **Mac**: Applications > Utilities > Terminal

In the window, you will see a cursor where you are able to type. Type the following and press return (or enter) at each step. Make sure all spaces and symbols are identical to the example below, including all spaces ( ) and dashes (-).

Substitute your GitHub username instead of jdoe:

```
git config --global user.name jdoe
```

and then: substitute the email address you used to create your GitHub account:

```
git config --global user.email johndoe@example.com
```

You can check settings with the following:

```
git config --list
```

Quit the Terminal after typing:

```
exit
```

### 6.3.4 Install the GitHub application

There are several options to clone your repository to your local machine. When getting started, we recommend using the GitHub application. This is freely available for download. Follow the default instructions for downloading and installing from the following:

- **Windows**: https://windows.github.com/.
- **Mac**: https://mac.github.com/.

### 6.3.5 Create a folder called *github* on your computer

Because you will use GitHub to collaborate with your team or request support from the OHI team, it is important you save files in places where the file path that is universal and not specific to your computer. When team members save files in different places, this will create a lot of problems when collaborating, particularly between Macs and Windows machines.

**Please create a folder called github in your root directory**. The file path for this folder will be:

- **Windows**: `Users\[User]\Documents\github\`
- **Mac**: `Users/[User]/github/`

This folder can be identified by any computer as `~/github/`.

> TIP: You can check the location of your `github` folder by right-clicking the folder icon and selecting 'Get Info' on a Mac or 'Properties' on Windows.

### 6.3.6 Clone your repository to your computer

Clone a repository by clicking the 'Clone in Desktop' button on your online repository's homepage (https://github.com/OHI-Science/{[}assessment{]}):

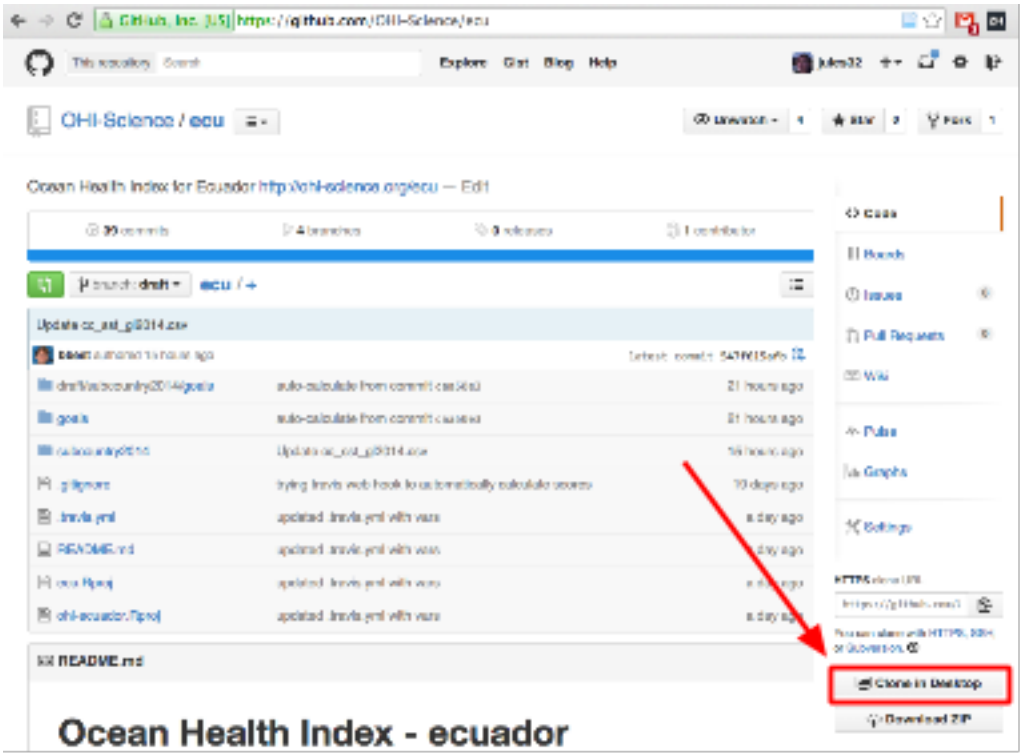



Figure 30:

You will be asked where to save this repository: save it into the `github` folder you created. The file path for your assessment will therefore be:

- Windows: C: `\Users\[User]\Documents\github\[assessment]` (example: C:\Users\johndoe\Documents\github
- on a Mac: `/Users/[User]/github/[assessment]` (example: `/Users/johndoe/github/ecu`)

The assessment can be identified by any computer as `~/github/[assessment]`.

The entire folder will now be saved on your computer.

### 6.3.7 Update permissions

You need to **email your username to ohi-science@nceas.ucsb.edu** for permission to upload modifications to your GitHub repository (you only need to do this once). Only team members who will be modifying files will need to do this; all other members can view online and download the repository without these permissions.

### 6.3.8 Work locally

You will then work locally on your own computer, modifying the files in the repository to reflect the desired modifications your team has identified for your assessment. Multiple users can work on the same repository at the same time, so there are steps involved to 'check in' your modifications so they can merge with the work of others without problems. GitHub has specific words for each of these steps. You have already successfully **cloned** an online repository to your local machine. After making modifications, you will **commit** these changes with a description before being able to sync back to the online repository. **Synching** involves both **pulling** any updates from the online repository before **pushing** committed changes back to the server.

> TIP: While you can edit files in the online GitHub repository, we do not recommend this. It is good practice to track changes through commits and syncing.

The example below illustrates GitHub's collaborative workflow with the `ohi-israel` repo owned by `OHI-Science`:

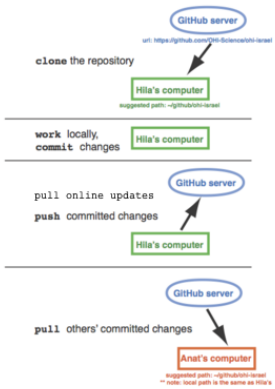



Figure 31:

**All changes within your local repository will be tracked by GitHub regardless of the software you use to make the changes**. This means that you can delete or paste files in the Mac Finder or Windows Explorer and edit *.csv* files in Excel or a text editor, and still sync these changes with the online repository. We recommend doing as much data manipulation as possible in a programming language like R, to maximize transparency and reproducibility. When modifying R scripts such as `functions.R`, you will need to work in R.

We recommend syncing with either the GitHub App or with RStudio. Both methods require you to commit your changes, before pulling any updates and pushing your modifications. The GitHub App combines the pulling and pushing into one step, called syncing. The following sections show you how to synchronize the repository on your computer with the repository online.

### 6.3.9 Syncing

When you work on your computer, any edits you make to any files in your repo, using any program, will be tracked by *git*. You can use any of the above to commit and sync your changes back to GitHub. There are many options you can use to sync your edits on a repo with the online version.

- **GitHub App** for Mac and for Windows
- **RStudio**
- **Command line**

If you are just modifying data *.csv* files, you probably only need to use the GitHub App. RStudio is convenient if you are working with *.R* files. Also, the command line can be used by those interested, and there are resources available online.

> TIP: Once you sync your repository, the updated information will be automatically available to the WebApps.

### 6.3.10 Using the GitHub App to synchronize your repository

The GitHub App will track your modifications and can be used to commit and sync any changes made locally to your repository. Once you are done working on the pertinent files and wish to commit and sync the changes to the online server on the Github server, open the GitHub App. The following example is with the `ecu` repository:

1. Make sure you select the correct repository, located on the left column of the GitHub App window (Step 1 in the figure).
2. Select the different files to which changes have been made (2a), and preview those changes on the right column of the GitHub App window (2b).
3. Once all the changes have been reviewed, write a summary/description in the respective message bars in the GitHub App window (3), then click on 'Commit' (3a) and then 'Sync' (3b) located on the top-right corner of the GitHub App window (Note: If a `Commit` button appears instead of `Commit & Sync`, you can either click `Commit` and then click the `Sync` in this way, or you can alternatively select *Edit* > *Automatically Sync After Committing* which will then allow you to click on 'Commit and Sync')

Go online and check that your changes are now visible on GitHub online.

### 6.3.11 Working with R and RStudio

**RStudio** is a program that can be used to synchronize any modifications you make to files in your assessment's repository, and if you are working in R, it is convenient since you do not need to open the GitHub App. If you do not already have this installed, install the latest version of R and RStudio (and if you do have these installed, check for updates: there are frequent updates to the R software, and the current version is identified on the website). Both R and RStudio are freely available to download.

**R**: Download the current version of R appropriate for your operating system at http://cran.r-project.org/ and follow the instructions to install it on your computer. If updating, compare the available version on their website with what you already have on your computer by typing sessionInfo() into your R console.

**RStudio**: Download the current version of RStudio software at www.rstudio.com. RStudio is not updated as often as R, but it is good to check for updates regularly. Note that in this case, you should follow the default install instructions.

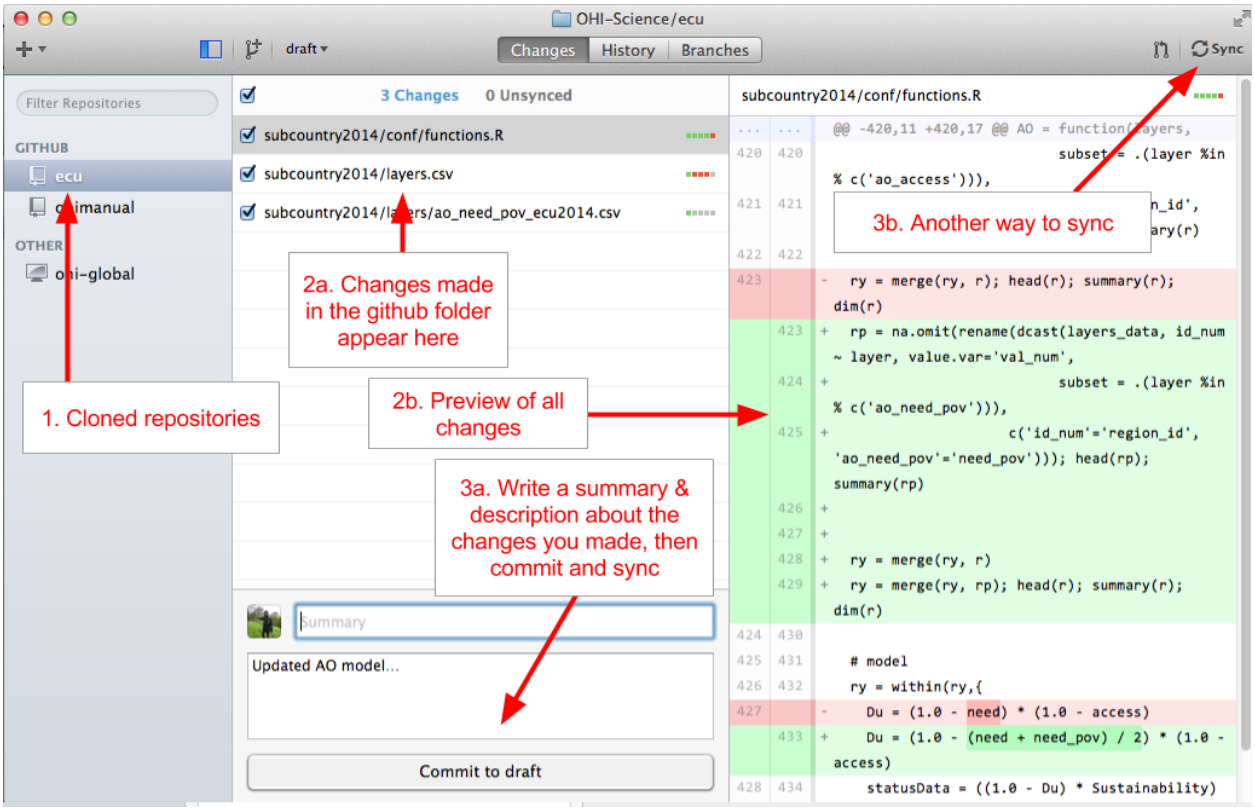



Figure 32: Figure showing the layout of the GitHub App when syncing. Click on 'Commit' and then 'Sync' to push changes to your repository.

If you are working on a Mac, you will need to tell RStudio to use the proper version of Git by doing the updating the preferences for 'Git executable':

**RStudio > Preferences... > Git/SVN > Git executable: /usr/local/git/bin/git**

### 6.3.12 Using RStudio to synchronize your repository

RStudio can sync files with GitHub directly, and can be used instead of the GitHub App. Like the GitHub App, it will capture the changes made to any files within the repository, no matter which software was used to modify them. The advantage for using RStudio to sync instead of the GitHub App is if you are working with R scripts already. In RStudio, you sync by first pulling and then pushing (separately); in the GitHub App these two functions are done together.

Launch your project in RStudio by double-clicking the `.Rproj` file in the assessment folder on your local hard drive.

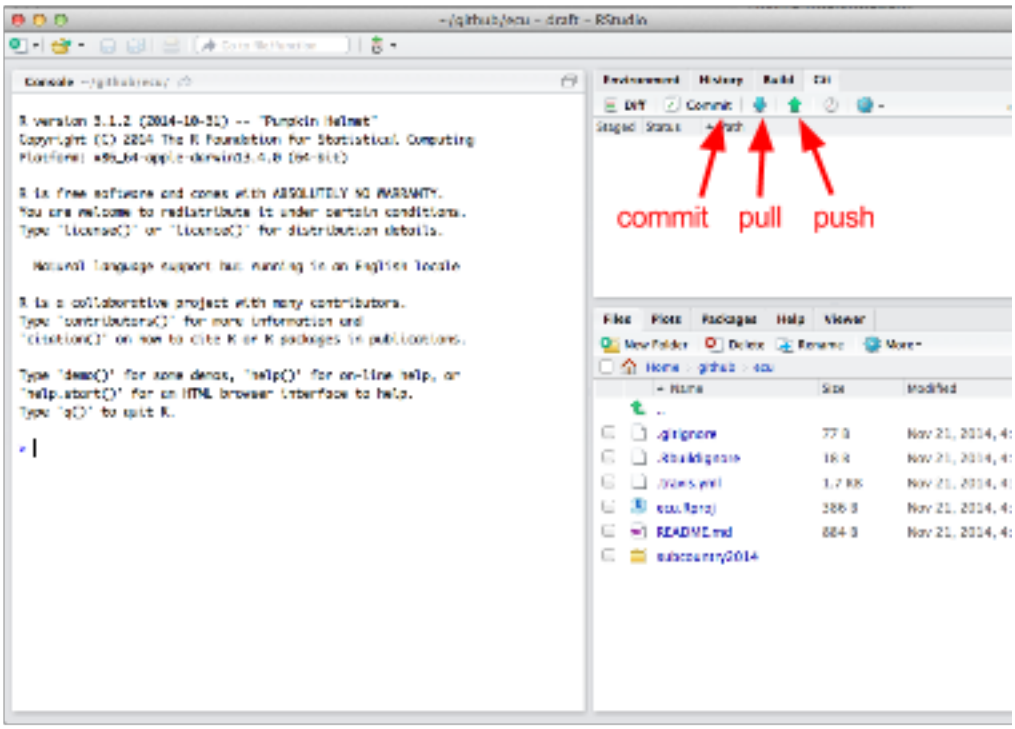



Figure 33:

When you modify or add a file, the file will appear in the 'Git' window once it has been saved. In the example below, the file `test.R` was created.

1. Clicking the 'Staged' box and the 'Commit' button opens a new window where you can review changes.
2. Type a commit message that is informative to the changes you've made.

- Note 1: there will often be multiple files 'staged' at the same time, and so the same commit message will be associated with all of the updated files. It is best to commit changes often with informative commit messages.

- Note 2: clicking on a staged file will identify additions and deletions within that file for your review

3. Click 'Commit' to commit the changes and the commit message
4. Pull any changes that have been made to the online repository. This is important to ensure there are no conflicts with updating the online repository.
5. Push your committed changes to the online repository. Your changes are now visible online.

   TIP: If you aren't seeing your changes in the 'Git' window, try saving the file again.

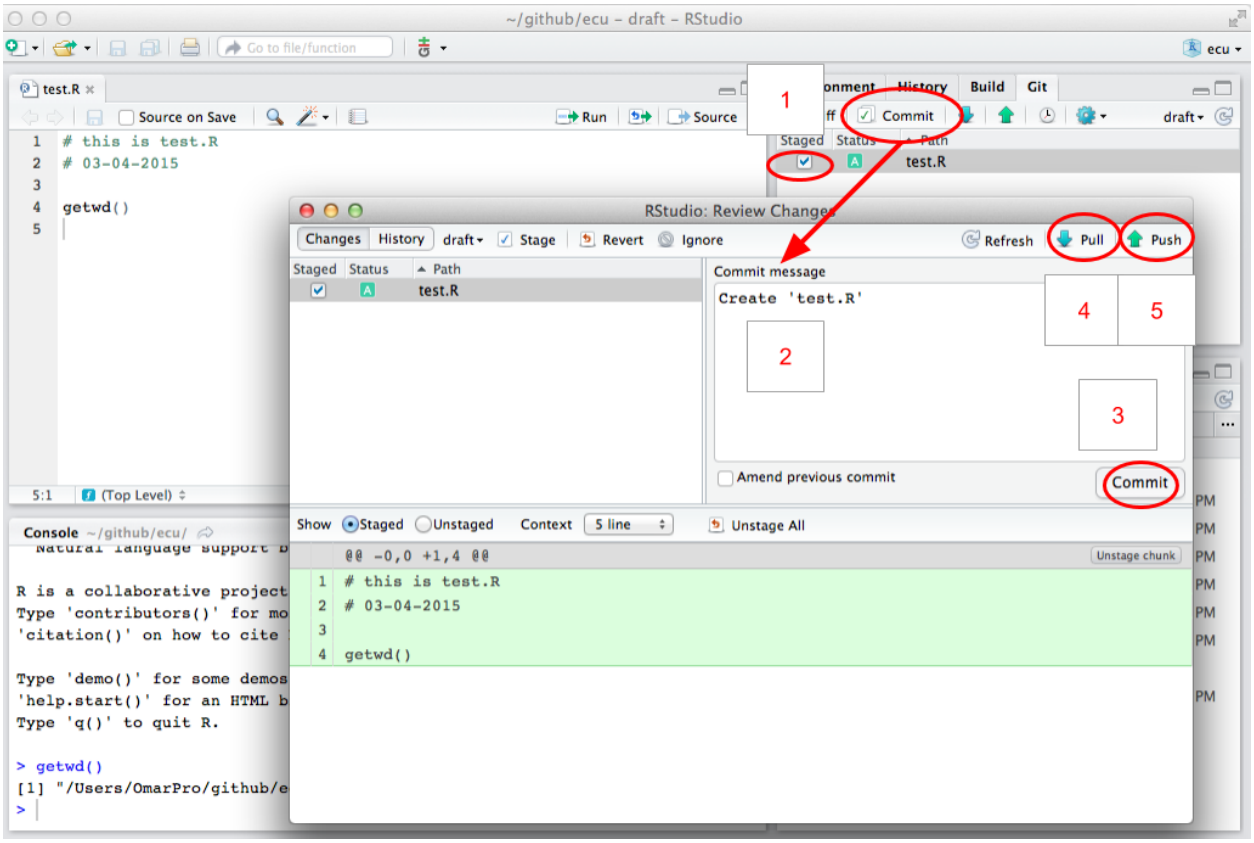



Figure 34: Figure showing RStudio when sycing. After first staging your changes, click the 'commit' button to open a new window where you can enter a 'commit message' and then pull and push new changes.

TIP: Another way to sync and open the project is to click on 'New Project' in the upper-right-hand corner of Rstudio, then choose 'Version Control', and then you can paste the URL of the desired repository. This URL can be found on on your online repository's homepage.

### 6.3.13 Install the latest version of R and RStudio

Make sure you have the most current version of R and RStudio. Download **R** at http://cran.r-project.org/ and install on your computer. If you already have R installed, check the website for updates. There are frequent updates to the R software, and the current version is identified on the website. Compare what is available from their website with what you already have on your computer by typing `sessionInfo()` into your R console. (This will also identify packages you have installed).

While not required, we highly recommend working with **RStudio**, which is an interface that makes working with R much easier, and it also interfaces with GitHub so you are able to synchronize without using the GitHub App. RStudio does not get updated as often as R does, but it is good to check for updates regularly.

### 6.4 GitHub repository architecture

GitHub stores all data files and scripts for your assessment in a repository (a folder). Different copies or complements to these folders, called *branches* can also exist, which aid with versioning and drafting. Your repository has four branches, two of which are displayed on your website (e.g., ohi-science.org/ecu):

1. **draft** branch is for editing. This is the default branch and the main working area where existing scenario data files can be edited and new scenarios added.
2. **published** branch is a vetted copy of the draft branch, not for direct editing. This branch is only updated by automatic calculation of scores if:
    1. no errors occur during the calculation of scores in the draft branch, and
    2. publishing is turned on. During the draft editing and testing phases of development, it is typically desirable to turn this off.
3. **gh-pages** branch is this website. The results sections of the site (regions, layers, goals, scores per branch/scenario) are overwritten into this repository after automatic calculation of scores. The rest of the site can be manually altered.
4. **app** branch is the interactive layer and map viewer application. The user interface and server-side processing use the Shiny R package and are deployed online via ShinyApps.io to your website. Once deployed, the WebApp pulls updates from the data branches (draft and published) every time a new connection is initiated (i.e., browser refreshes).

    TIP: When looking at files on GitHub, note that the timestamps are associated with the 'commit' time rather than the 'push' time.

# 7 Using the Toolbox

**Section Summary:**

In this section, you will learn about the most common modifications made to repositories. You will be given examples to follow to help with your own assessment. The most common modifications are changing the pressures and resilience matrices, changing or creating data layers, and changing or removing goals models.

TIP: You should have access to your assessment repository and be familiar with the files in the folder.

As your team finalizes which data should be included in the assessment and begins developing goal models, you can incorporate this information into your repository. Input information must be properly formatted into **layers**, which are registered with the Toolbox for use. Layers for the Toolbox can be prepared with any software that handles *.csv* files, but goal models must be updated in R. It is recommended that layer preparation occurs within your repository's `prep` folder as much as possible, as it will also be archived by GitHub. Calculations can be done locally and offline by running `subcountry2014/calculate_scores.R`.

### 7.1 Layer preparation workflow

It is recommended that you construct a useful workflow with your team to incorporate local information into the Toolbox. Adding layers to the Toolbox will require working with GitHub and the Toolbox file system structure. The overall process involves preparing the layers (which can be done in the `prep` folder), saving them in the `layers` folder, and registering the layers. The layer preparation process can occur in tandem with the model modification process.

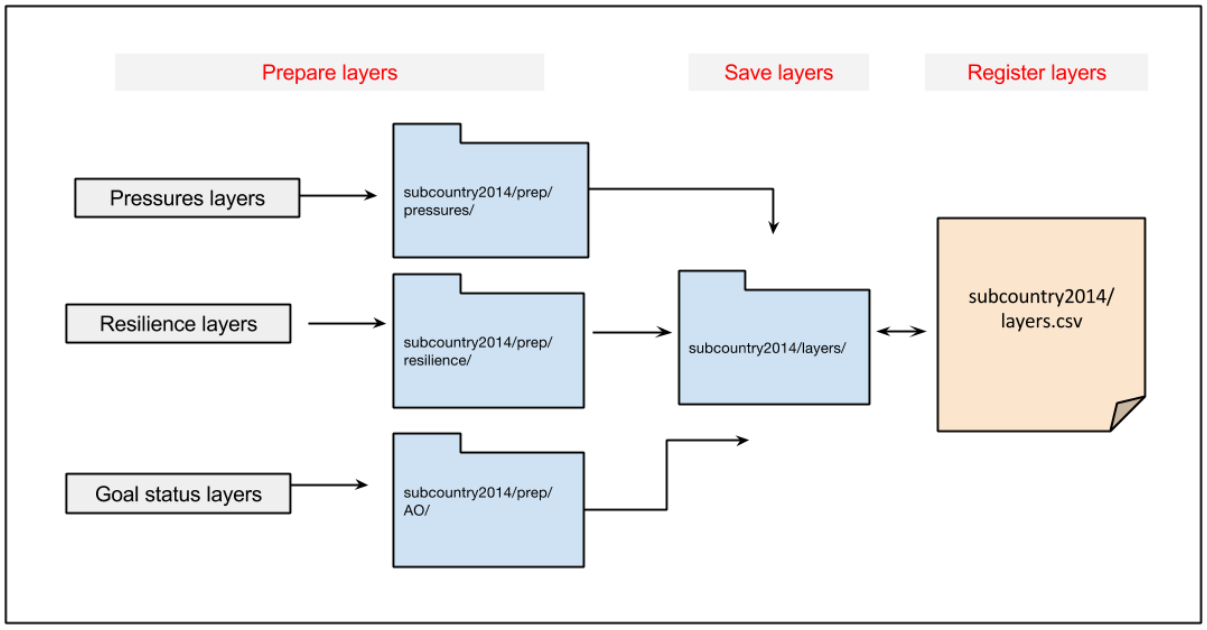



Figure 35: Diagram of OHI Toolbox data preparation workflow. You should start by prepping the files, loading them into the `layers` folder when they're ready for the Toolbox, and then registering them in `layers.csv`

## 7.2 Modifying and creating data layers

Data layers are *.csv* files and are located in the `[assessment]/subcountry2014/layers` folder. Remember that all data layers provided in your repository are extracted from the global 2014 assessment.

- Layers with the suffix `_gl2014.csv` (*gl* for *global*) have been exactly copied from the global assessment and applied equally to each region, and therefore the values will be the same across all subcountry regions.
- Layers with the suffix `_sc2014.csv` (*sc* for *subcountry*) have been spatially-extracted from global data or adjusted with spatially-extracted data so that each region in your assessment has a unique value. For example, gross domestic product (GDP) used in the global assessment was reported at the national (most often country) level. Instead of being applied equally across all subcountry regions (which would incorrectly increase the nation's GDP several times), national GDP was down-weighted by the proportion of coastal population in each region compared with the total coastal population.

Both types of default data layers are of coarse-resolution and should be replaced with local, high-resolution data when possible. The priority should be to replace as much of the `_gl2014.csv` data as possible.

**There are several steps to follow when working with data layers:**

1. Modify or create data layer with proper formatting
2. Save the layer in the `layers` folder
3. Register the layer in `layers.csv`
4. Check (and update when appropriate) `pressures_matrix.csv` and `resilience_matrix.csv` (located in: `[assessment]/subcountry2014/conf`)

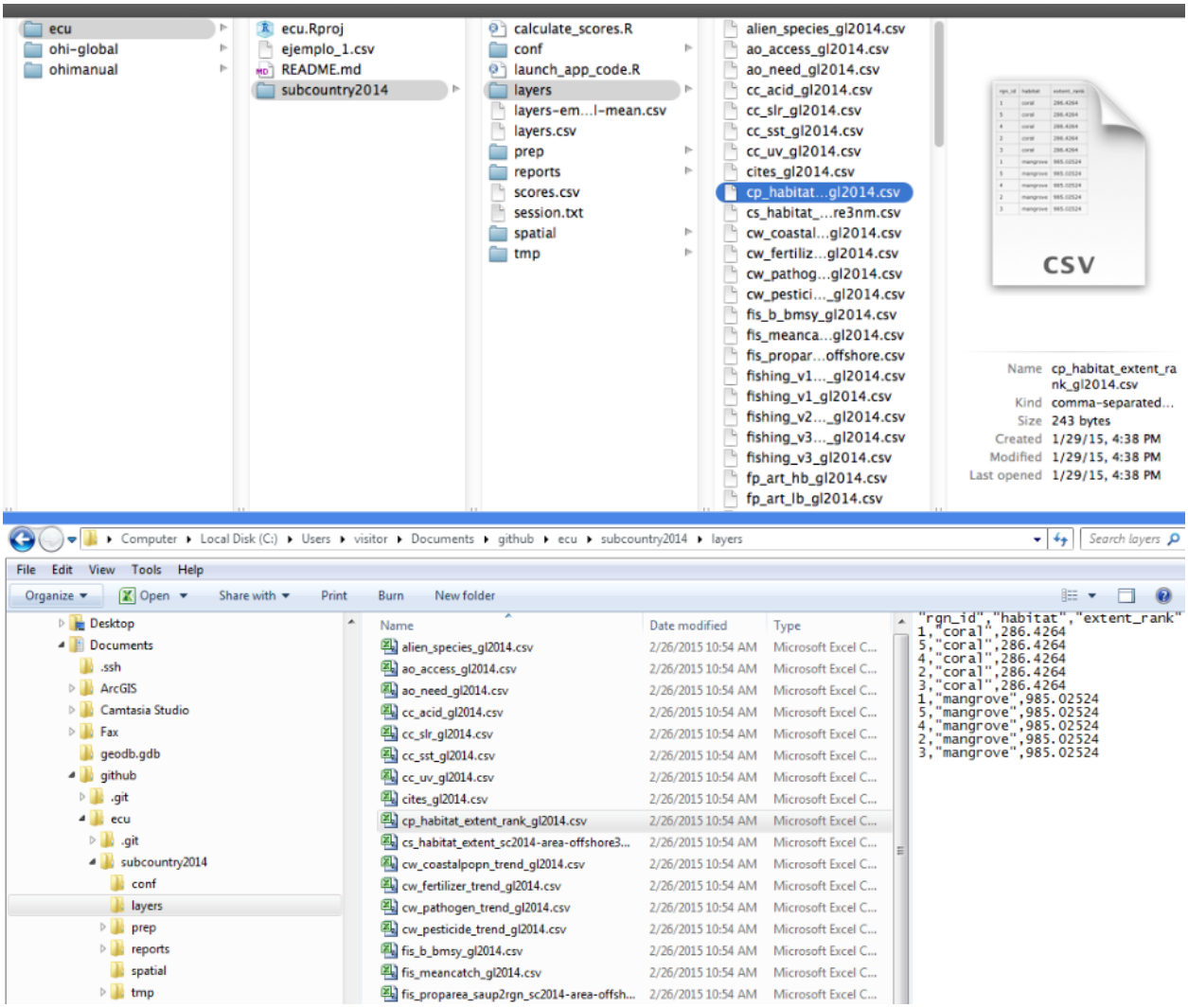



Figure 36: This figure shows the location of your data layers. Mac navigation is shown above and Windows is shown below.

### 7.2.1 Create data layers with proper formatting

The OHI Toolbox expects each data layer to be in its own *.csv* file and to be in a specific format, with data available for every region within the study area, with data organized in 'long' format (as few columns as possible), and with a unique region identifier (*rgn_id*) associated with a single score or value. See the **'Formatting data for the Toolbox'** section for more information.

### 7.2.2 Save data layers in the *layers* folder

When you modify existing or create new data layers, we recommend saving this as a new *.csv* file with a suffix identifying your assessment (example: `_sc2014.csv`). Modifying the layer name provides an easy way to track which data layers have been updated regionally, and which rely on global data. Then, the original layers (`_gl2014.csv` and `_sc2014.csv`) can be deleted.

* Note: filenames should not have any spaces: use an underscore ('_') instead. This will reduce problems when R reads the files.

### 7.2.3 Register data layers in `layers.csv`

When there are new filenames associated with each layer, they will need to be registered in `[assessment]/subcountry2014/layers.csv`. If a layer simply has a new filename, only the *filename* column needs to be updated:

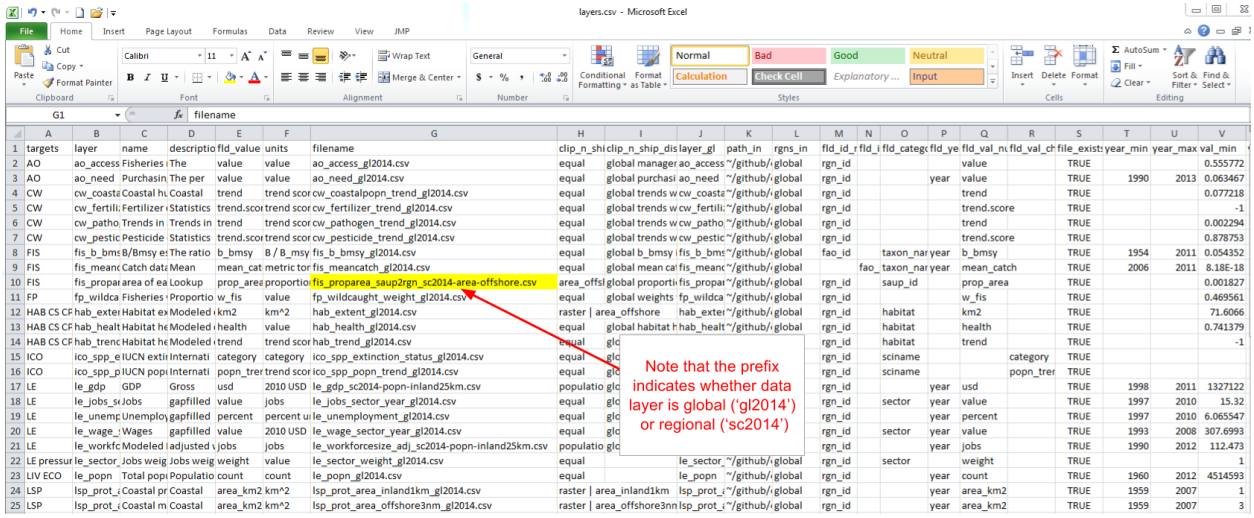



Figure 37: Register new layers in `layers.csv`. Be sure to note if there is a change in the filename.

> TIP: This part is done manually. If you prefer not to manipulate your file by hand, you can generate a script that automates this.

However, if a new layer has been added (for example when a new goal model is developed), you will need to add a new row in the registry for the new data layer and fill in the first eight columns (columns A-H). It is important to check that you have filled you the fields correctly, for instance, if "fld_value" does not match the header of the source data layer, you will see an error message when you try to calculate scores. Other columns are generated later by the Toolbox as it confirms data formatting and content:

- **targets:** Add the goal/dimension that the new data layer relates to. Goals are indicated with two-letter codes and sub-goals are indicated with three-letter codes, with pressures, resilience, and spatial layers indicated separately.
- **layer:** Add an identifying name for the new data layer, which will be used in R scripts like `functions.R` and *.csv* files like `pressures_matrix.csv` and `resilience_matrix.csv`.
- **name:** Add a longer title for the data layer–this will be displayed on your WebApp.
- **description:** Add a longer description of the new data layer–this will be displayed on your WebApp.
- **fld_value:** Add the appropriate units for the new data layer (which will be referenced in subsequent calculations).
- **units:** Add a description about the *units* chosen in the *fld_value* column above.
- **filename:** Add a filename for the new data layer that matches the name of the *.csv* file that was created previously in the `layers` folder.
- **fld_id_num:** Area designation that applies to the newly created data layer, such as: *rgn_id* and *fao_id*.

TIP: Think about what units you would like to be displayed on the WebApp when filling out "units."

#### 7.2.4 Check pressures and resilience matrices

If the new or modified layer is a pressures layer, check that `pressures_matrix.csv` and `resilience_matrix.csv` have been properly modified to register the new data layers.

### 7.3 Modifying pressures matrices

Your team will identify if any pressures layers should be added to the pressures matrices, and if so, which goals the pressure affects and what weight they should have. You can transfer this information in `pressures_matrix.csv` (located in the `[assessment]/subcountry2014/conf` folder). It is important to note that the matrix identifies the pressures relevant to each goal, and which weight will be applied in the calculation. Each pressure is a data layer, located in the `subcountry2014/layers` folder. This means that pressure layers need information for each region in the study area, and some layers will need to be updated with local data. In modifying pressures, you will need to consider whether data layers can be updated or added, and whether data layers map onto goals appropriately in the local context.

Adding a new pressure to the pressures matrix requires the following steps:

1. Create new pressure layer(s) and save in the `layers` folder
2. Register pressure layer(s) in `layers.csv`
3. Register pressure layer(s) in `pressures_matrix.csv`

- a. Set the pressure category

- b. Identify the goals affected and set the weighting
- c. Modify the resilience matrix (if necessary)

The following is an example of adding two new pressures layers.

### 7.3.1 Create the new pressure layers and save in the `layers` folder

If you create a new data layer, give it a short but descriptive name that also includes a prefix that signifies the pressure category (for example: *po_* for the pollution category). There are five physical categories and one social category:

- *po_* = pollution
- *hd_* = habitat destruction
- *fp_* = fishing pressure
- *sp_* = species pollution
- *cc_* = climate change
- *ss_* = social pressure

So for example, `po_trash` is a pollution layer with trash on beaches, and `sp_alien` is species pollution due to alien (invasive) species.

In the current example, the two new layers created to account for the input and output effects of desalination operations will be called *po_desal_in*, and *po_desal_out*.

These new layers will have scores from 0 to 1, with values for each region in your study area, and will be saved in the `layers` folder.

### 7.3.2 Register the new pressure layers in `layers.csv`

Add two new rows in `layers.csv`, and register the new pressure layers by filling out the first eight columns for *po_desal_in*, and *po_desal_out*.

layers.csv

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | targets | layer | name | description | fld_value | units | filename | fld_id_num |
| 60 | TR | tr_sustainabili | Sustainability in | Tourism Comp | score | score | tr_sustainability_global2013.cs | rgn_id |
| 61 | TR | tr_unemployr | Percent unemployment | | percent | percent unemplc | tr_unemployment_global2013. | rgn_id |
| 62 | pressures | po_desal_in | Example data | Made-up data | value | pressure score | po_desal_in_china2014.csv | rgn_id |
| 63 | pressures | po_desal_out | Example data | Made-up data | value | pressure score | po_desal_out_china2014.csv | rgn_id |
| 64 | pressures | cc_acid | Ocean acidificati | Modeled distri | pressure_scor | pressure score | cc_acid_global2013.csv | rgn_id |
| 65 | pressures | cc_slr | Sea level rise | Modeled sea le | pressure_scor | pressure score | cc_slr_global2013.csv | rgn_id |

Figure 38:

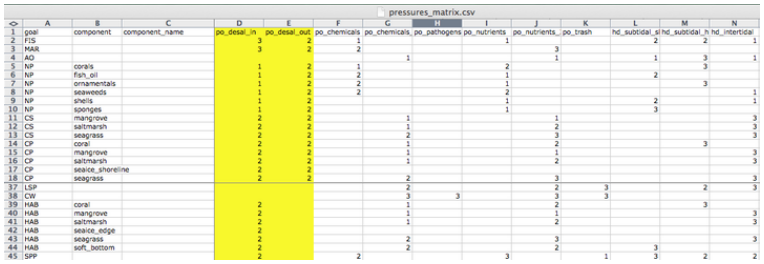
### 7.3.3 Register the new layers in `pressure_matrix.csv`

`pressures_matrix.csv` identifies the different types of ocean pressures (columns) with the goals that they affect (rows). Adding a new pressures layer to `pressures_matrix.csv` requires adding a new column with the pressure layer name.

**7.3.3.1 Set the pressure category** This step requires transferring previous decisions made by your team into `pressures_matrix.csv`. Each pressure category is calculated separately before being combined with the others, so it is important to register the new pressure with the appropriate category prefix decided by your regional assessment team.

**7.3.3.2 Identify the goals affected and set the weighting** This step also requires transferring prior decisions into `pressures_matrix.csv`. Mark which goals are affected by this new pressure, and then set the weighting. Pressures weighting by goal should be based on scientific literature and expert opinion (3 = highly influential pressure, 2 = moderately influential pressure, 1 = not very influential pressure). Remember that the rankings in the pressures matrix are separate from the actual data within the pressures data layers. The rankings ensure that within a particular goal (e.g. within a row of the pressures matrix), the stressors that

more strongly influence the goal's delivery have a larger contribution to that goal's overall pressure score. Therefore, the rankings are assigned independently of the actual pressure scores, and only determine their importance within the calculations.



pressures_matrix.csv

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | goal | component | component_name | po_desal_in | po_desal_out | po_chemicals | po_chemicals | po_pathogens | po_nutrients | po_nutrients | po_trash | hd_subtidal_sl | hd_subtidal_h | hd_intertidal |
| 2 | FIS | | | 3 | 2 | 1 | | | 1 | | | 2 | 2 | 1 |
| 3 | MAR | | | 3 | 2 | 2 | | | | 3 | | | | |
| 4 | AO | | | | | | 1 | | | 1 | | 1 | 3 | 1 |
| 5 | NP | corals | | 1 | 2 | 1 | | | 2 | | | | 3 | |
| 6 | NP | fish_oil | | 1 | 2 | 2 | | | 1 | | | 2 | | |
| 7 | NP | ornamentals | | 1 | 2 | 2 | | | 1 | | | | 3 | |
| 8 | NP | seaweeds | | 1 | 2 | 2 | | | 2 | | | | | 1 |
| 9 | NP | shells | | 1 | 2 | | | | 1 | | | 2 | | 1 |
| 10 | NP | sponges | | 1 | 2 | | | | 1 | | | 3 | | |
| 11 | CS | mangrove | | 2 | 2 | | 1 | | | 1 | | | | 3 |
| 12 | CS | saltmarsh | | 2 | 2 | | 1 | | | 2 | | | | 3 |
| 13 | CS | seagrass | | 2 | 2 | | 2 | | | 3 | | | | 3 |
| 14 | CP | coral | | 2 | 2 | | 1 | | | 2 | | | 3 | |
| 15 | CP | mangrove | | 2 | 2 | | 1 | | | 1 | | | | 3 |
| 16 | CP | saltmarsh | | 2 | 2 | | 1 | | | 2 | | | | 3 |
| 17 | CP | seaice_shoreline | | 2 | 2 | | | | | | | | | |
| 18 | CP | seagrass | | 2 | 2 | | 2 | | | 3 | | | | 3 |
| 37 | LSP | | | | | | 2 | | | 2 | 3 | | 2 | 3 |
| 38 | CW | | | | | | 3 | 3 | | 3 | 3 | | | |
| 39 | HAB | coral | | 2 | | | 1 | | | 2 | | | 3 | |
| 40 | HAB | mangrove | | 2 | | | 1 | | | 1 | | | | 3 |
| 41 | HAB | saltmarsh | | 2 | | | 1 | | | 2 | | | | 3 |
| 42 | HAB | seaice_edge | | 2 | | | | | | | | | | |
| 43 | HAB | seagrass | | 2 | | | 2 | | | 3 | | | | 3 |
| 44 | HAB | soft_bottom | | 2 | | | 2 | | | 2 | | 3 | | |
| 45 | SPP | | | 2 | | 2 | | | 3 | | 1 | 3 | 2 | 2 |

Figure 39:

### 7.3.4 Modify the resilience matrix (if necessary)

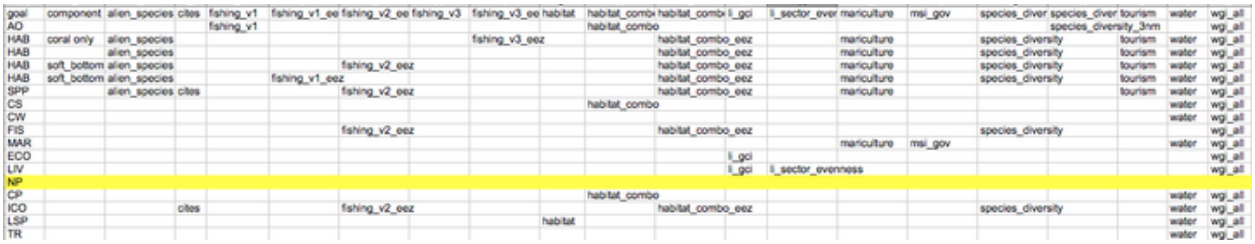
Resilience is included in OHI as the sum of the ecological factors and social initiatives (policies, laws, etc.) that can positively affect goal scores by reducing or eliminating pressures. The addition of new pressure layers may therefore warrant the addition of new resilience layers that were not previously relevant. Similarly, the removal of pressure layers may warrant the removal of now irrelevant resilience layers.

## 7.4 Modifying resilience matrices

Previous decisions made with your team will identify if any resilience layers should be added to the resilience matrices, and if so, which goals and/or pressures the resilience affects and what weight they should have. You can then transfer this information into `resilience_matrix.csv` (located in the `[assessment]/subcountry2014/conf` folder).

`resilience_matrix.csv` maps the different types of resilience (columns) with the goals that they affect (rows). New resilience layers may be added to `resilience_matrix.csv` based on finer-scale local information either in response to a new pressures layer, or as a new independent measure. Any added layer must be associated with a pressures layer that has a weight of 2 or 3 in the OHI framework so that resilience measures can mitigate pressures in each region.

Each goal must have a resilience measure associated with it. In the figure below, the Toolbox would give an error because there are no resilience layers indicated for the natural products (NP) goal.



| goal | component | alien_species | cites | fishing_v1 | fishing_v1_eez | fishing_v2_eez | fishing_v3 | fishing_v3_eez | habitat | habitat_combo | habitat_combo | li_gci | li_sector_ever | mariculture | msi_gov | species_diver | species_diver | tourism | water | wgi_all |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AO | | | | fishing_v1 | | | | | | habitat_combo | | | | | | | species_diversity_3nm | | | wgi_all |
| HAB | coral only | alien_species | | | | | | fishing_v3_eez | | | habitat_combo_eez | | | mariculture | | species_diversity | | tourism | water | wgi_all |
| HAB | | alien_species | | | | | | | | | habitat_combo_eez | | | mariculture | | species_diversity | | tourism | water | wgi_all |
| HAB | soft_bottom | alien_species | | | | fishing_v2_eez | | | | | habitat_combo_eez | | | mariculture | | species_diversity | | tourism | water | wgi_all |
| HAB | soft_bottom | alien_species | | | fishing_v1_eez | | | | | | habitat_combo_eez | | | mariculture | | species_diversity | | tourism | water | wgi_all |
| SPP | | alien_species | cites | | | fishing_v2_eez | | | | | habitat_combo_eez | | | mariculture | | | | tourism | water | wgi_all |
| CS | | | | | | | | | | habitat_combo | | | | | | | | | water | wgi_all |
| CW | | | | | | | | | | | | | | | | | | | water | wgi_all |
| FIS | | | | | | fishing_v2_eez | | | | | habitat_combo_eez | | | | | species_diversity | | | | wgi_all |
| MAR | | | | | | | | | | | | | | mariculture | msi_gov | | | | water | wgi_all |
| ECO | | | | | | | | | | | | li_gci | | | | | | | | wgi_all |
| LIV | | | | | | | | | | | | li_gci | li_sector_evenness | | | | | | | wgi_all |
| NP | | | | | | | | | | | | | | | | | | | | |
| CP | | | | | | | | | | habitat_combo | | | | | | | | | water | wgi_all |
| ICO | | | cites | | | fishing_v2_eez | | | | | habitat_combo_eez | | | | | species_diversity | | | water | wgi_all |
| LSP | | | | | | | | | habitat | | | | | | | | | | water | wgi_all |
| TR | | | | | | | | | | | | | | | | | | | water | wgi_all |

Figure 40:

### 7.4.1 Updating resilience matrix with local habitat information

In this example we will borrow from the experience of `ohi-israel`, where they assessed habitats in the Habitats (HAB) sub-goal that were not included in global assessments `ohi-global`. Therefore, the resilience matrix needed some revision.

The habitats assessed for `ohi-israel` are:

> `rocky_reef`, `sand_dunes`, `soft_bottom`

Updates are required for the following files:

- *layers.csv*
- *resilience_matrix.csv*
- *resilience_weights.csv* (only if adding new resilience layers)

#### 7.4.1.1 Global resilience layers

The first step is to determine which resilience layers from the global assessment are relevant to your assessment, and whether others need to be added. The full list of layers included in the global resilience matrix are:

> `alien_species`, `cites`, `fishing_v1`, `fishing_v1_eez`, `fishing_v2_eez`, `fishing_v3`, `fishing_v3_eez`, `habitat`, `habitat_combo`, `habitat_combo_eez`, `li_gci`, `li_sector_evenness`, `mariculture`, `msi_gov`, `species_diversity`, `species_diversity_3nm`, `tourism`, `water`, `wgi_all`

Some of these layers capture general aspects of governance that apply to the protection of any habitat. These are:

> `alien_species`, `cites`, `msi_gov`, `water`, `wgi_all`

Two layers only apply to the livelihoods and economies goal (LE), so they should be excluded from HAB resilience:

> `li_gci`, `li_sector_evenness`

The remaining layers apply to certain habitats, but not others. We focus on these to determine how to adapt the HAB resilience calculation for `ohi-israel`. They are:

> `fishing_v1`, `fishing_v1_eez`, `fishing_v2_eez`, `fishing_v3`, `fishing_v3_eez`, `habitat`, `habitat_combo`, `habitat_combo_eez`, `mariculture`, `species_diversity`, `species_diversity_3nm`, `tourism`

#### 7.4.1.2 Determining how to modify these resilience layers

- To determine whether `species_diversity_3nm` or `species_diversity` should be used:
  - `sand_dunes` should use `species_diversity_3nm`,
  - `soft_bottom` should use `species_diversity`,
  - is `rocky_reef` mainly coastal? if so it should use `tourism` and `species_diversity_3nm`.
- If the habitats can be affected by mariculture plants (e.g. eutrophication and decreased water quality can occur if mariculture plants are close by and have poor wastewater treatment), then the `mariculture` resilience score should be added.
  - are there any mariculture plants in Israel? If yes, on which habitats do they occur?
- The remaining layers are the `fishing_v...` and `habitat..` layers, which are composite indicators obtained from different combinations of the following indicators:

`Mora, Mora_s4, CBD_hab, MPA_coast, MPA_eez`,

where:

- `Mora` is a fisheries governance effectiveness indicator by Mora *et al* (2009)
- `Mora_s4` is another indicator from Figure S4 of the supplementary material of the same publication that focuses on regulations of artisanal and recreational fisheries
- `CBD_hab` is a score assigned based on answers to a questionnaire compiled by countries that committed to Rio's Convention on Biological Diversity (CBD) to establish their progress towards habitat biodiversity protection
- `MPA_coast` is an indicator obtained as the proportion of coastal (3nm) waters that are in a marine protected area (MPA), with the maximum being 30% of coastal waters
- `MPA_eez` is an indicator obtained as the proportion of the whole EEZ that is in a marine protected area, with the maximum being 30% of the whole EEZ.

This table shows which indicators are used by each combo layer:

| Layer | Mora | Mora_s4 | CBD_hab | MPA_coast | MPA_eez |
|---|---|---|---|---|---|
| fishing_v1 | Mora | | CBD_hab | MPA_coast | |
| fishing_v1_eez | Mora | | CBD_hab | | MPA_eez |
| fishing_v2_eez | Mora | Mora_s4 | CBD_hab | | MPA_eez |
| fishing_v3 | | Mora_s4 | CBD_hab | MPA_coast | |
| fishing_v3_eez | | Mora_s4 | CBD_hab | | MPA_eez |
| habitat | | | CBD_hab | | |
| habitat_combo | | | CBD_hab | MPA_coast | |
| habitat_combo_eez | | | CBD_hab | | MPA_eez |

**Questions to consider**:

The first objective is to determine whether the general `fishing_v..` or `habitat_...` categories are relevant to each of the habitats. For example, fisheries regulations do not affect the conservation of sand dunes, so this habitat should not use any of the fisheries combos. If the general resilience categories are relevant to the habitat, the next step is to select one resilience layer within the `fishing_v...` and `habitat...` categories that most adequately captures the suite of combined resilience variables that affect the habitat. For example, the sand dune habitat is a strictly coastal habitat, so the most appropriate resilience layer would be the one that uses the MPA_coast (i.e., `habitat_combo`). The rocky reef and soft bottom, on the other hand, should definitely include fisheries and habitat regulations. So, you'll need to choose a fisheries and a habitat combo for these two habitats. To do so, consider:

1) For which habitats should you use both a fishery and a habitat combo, or just use a habitat combo?

- fisheries regulations do not affect the conservation of sand-dunes, so this habitat should not use any of the fisheries combos. Also, this is a strictly coastal habitat, so choose the habitat layer that uses the `MPA_coast` instead of the `MPA_eez`, i.e. `habitat_combo` (and, as mentioned above, choose the coastal version of biodiversity, i.e. `species_diversity_3nm`).
- The rocky reef and soft bottom, on the other hand, should definitely include fisheries regulations. So you'll need to choose a fisheries and a habitat combo for these two habitats.

2) Which fisheries and habitat combos for `rocky_reef` and `soft_bottom`? The choice depends on two things:

- whether they are coastal habitats (within 3nm of the coast) or EEZ-wide habitats
  - if coastal, use the fisheries and habitat combos with `MPA_coast` (`fishing_v1`, `fishing_v3`, `habitat_combo`), and the `species_diversity_3nm` layer
  - if EEZ-wide, use the fisheries and habitat combos with `MPA_eez` (`fishing_v1_eez`, `fishing_v2_eez`, `fishing_v3_eez`, `habitat_combo_eez`), and the `species_diversity` layer
- whether the fisheries occurring on that habitat are mainly artisanal, mainly commercial, or both
  - if only commercial fisheries, use a layer that only uses the `Mora` data `fishing_v1..`)
  - if only artisanal/small-scale fisheries, use a layer that only uses the `Mora_s4` data (`fishing_v3..`)
  - if both, use a layer that uses both `Mora` and `Mora_s4` data (`fishing_v2..`)

3) It may also be that the existing global combo layers are not appropriate for your habitats. For example, if rocky reef is mainly coastal, and it is fished by both commercial and artisanal methods, then we need a new combo that uses `Mora`, `Mora_s4`, `CBD_hab`, and `MPA_coast` (this is the same as `fishing_v2_eez`, but we use the `MPA_coast` layer instead of the `MPA_eez`). All other combinations are already present.

4) Another issue to consider is whether local data are available to improve the pressure layers (that are based on global data). For example, if there are local data on Marine Protected Areas (MPAs) and any areas with special regulations, this should be used to generate the `MPA_coast` and `MPA_eez` layers.You may know that only certain types of protected areas are closed to fisheries, and may want to only include those. Also, local datasets may be more accurate and regularly updated. **NOTE: in the global study, these are the same datasets used to calculate the status of Lasting Special Places (LSP).

5) How to update `resilience_matrix.csv`?

- write the complete list of layers you want to use for each habitat. Based on the above, for example, `soft bottom` in Israel matches the combination of layers called *soft bottom, with corals* in the default `resilience_matrix.csv`. But the `rocky_reef` and `sand_dunes` don't seem to match any existing combination, so you'll probably need to delete some of the rows, e.g. the *coral only*, and replace with new ad-hoc rows.

## 7.5 Modifying goal models

When an existing layer is updated with new data, the Toolbox will automatically incorporate it into the goal calculations after the updated filenames are registered in `layers.csv`. However, if a new layer has been added to the layers folder and registered in `layers.csv`, the Toolbox will not use it unless it is called in a goal model. To integrate any new data layers registered in `layers.csv` you will need to modify the goal model to incorporate the data. Furthermore, in many cases, it will make sense to modify goal models based on data availability and/or local context. For example, the models for regional analyses can often be simplified because of improved data.

**There are some key steps to follow when working with goal models:**

1. Update `functions.R`
2. Check and possibly update `goals.csv`
3. Check if you need to update `pressures_matrix.csv` and `resilience_matrix.csv` when you change a goal model.

### 7.5.1 Update *functions.R*

To incorporate a new data layer into a goal model, open `functions.R` in RStudio: this script contains all the models for each goal and sub-goal. A member of your team with the ability to write R code will need to translate the updated goal model into the Toolbox format. Follow the structure of existing goal models in order to incorporate the new data layers, noting the use of certain R packages for data manipulation.

The image below shows the navigation pane in RStudio that can be used to easily navigate between goal models.

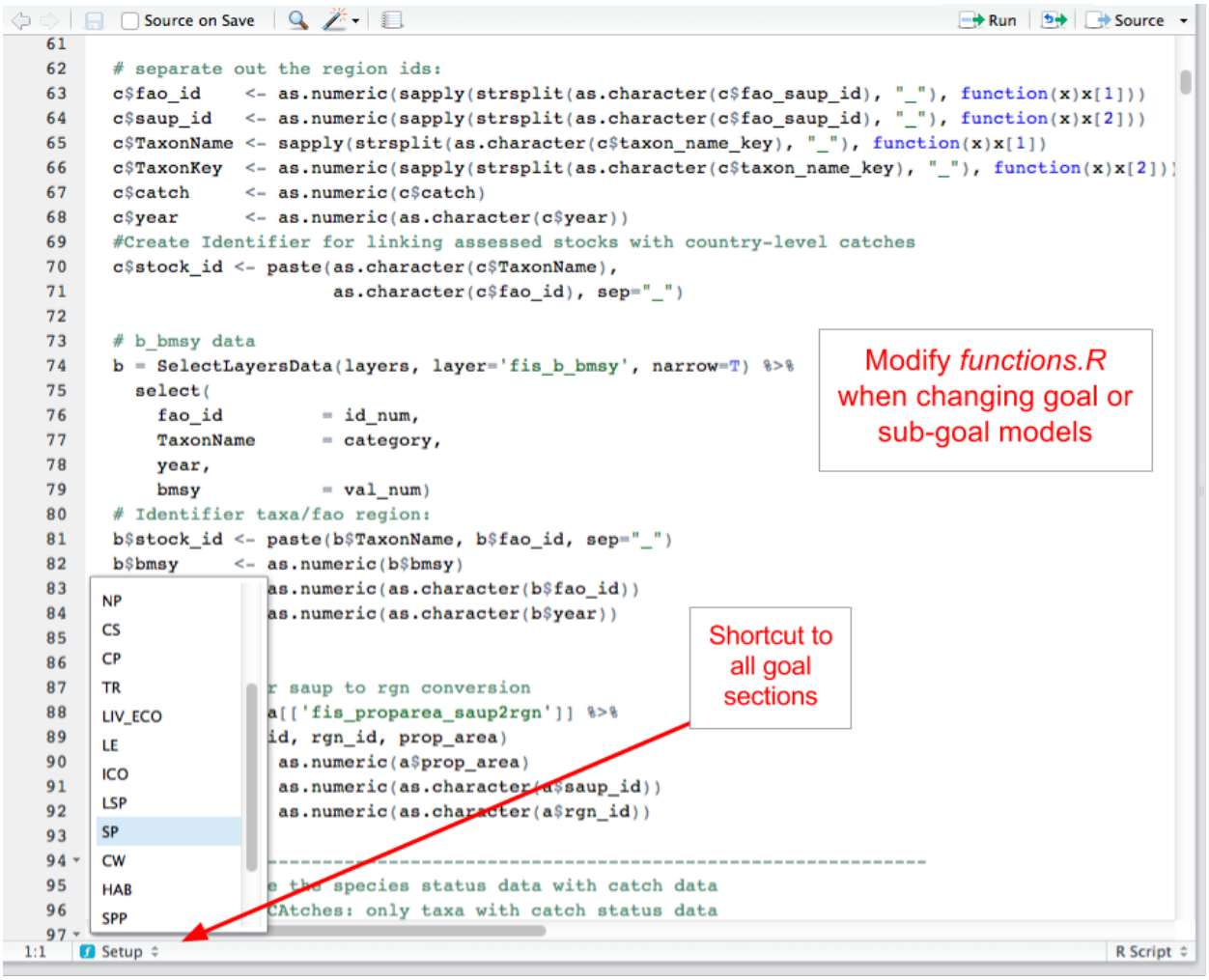



Figure 41: The navigation pane in RStudio can be used to easily navigate between goal models.

### 7.5.2 Check and possibly update *goals.csv*

`goals.csv` provides input information for `functions.R`, particularly about goal weighting and function calls. It also includes descriptions about goals and sub-goals, which is presented on the WebApp.

Changing goal weights will be done here by editing the value in the *weight* column. Weights do not need to be 0-1 or add up to 10; weights will be scaled as a proportion of the number of goals assessed. `goals.csv` also indicates the arguments passed to `functions.R`. These are indicated by two columns: *preindex_function* (functions for all goals that do not have sub-goals, and functions for all sub-goals) and *postindex_function* (functions for goals with sub-goals).

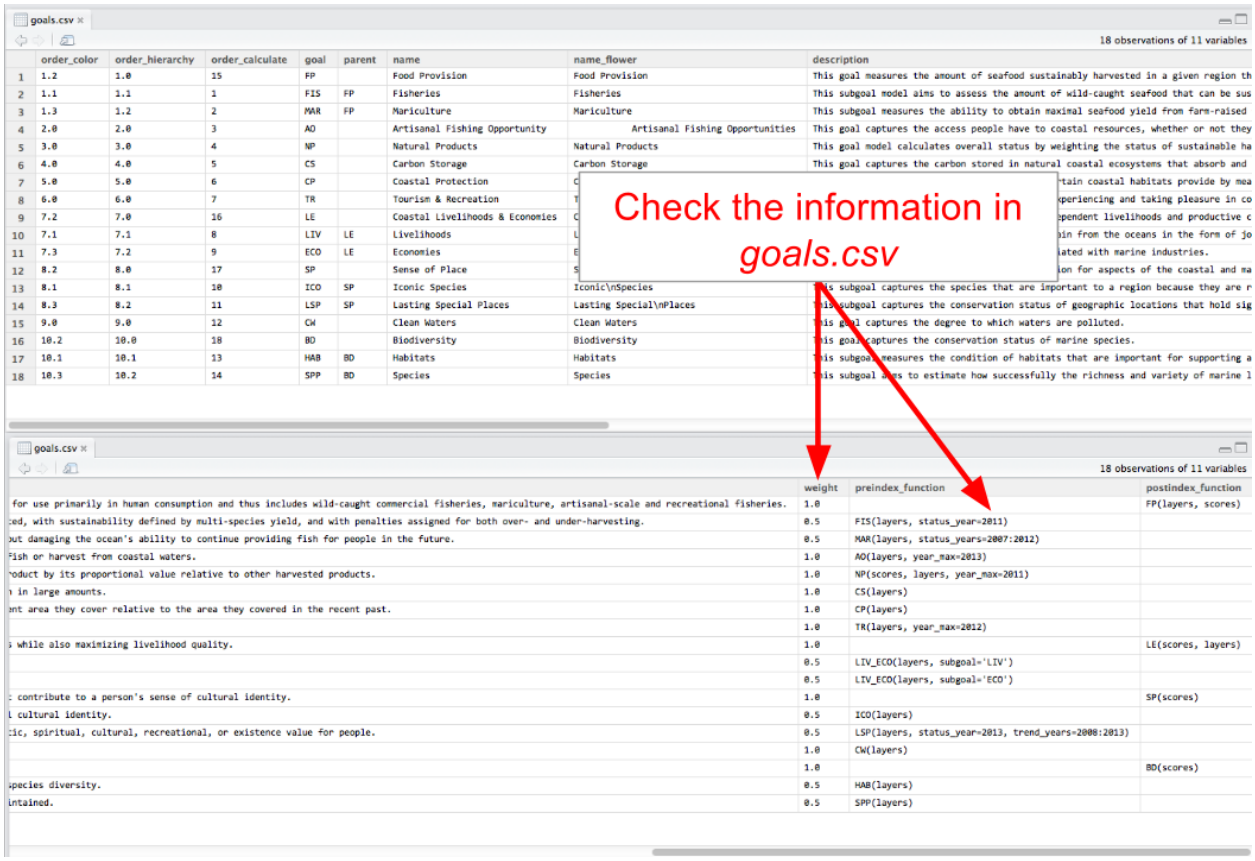



Figure 42: Check the information in `goals.csv`. It provides input information for `functions.R`.

**When updating layers or goal models, it is important to ensure that information called from `goals.csv` is correct**:

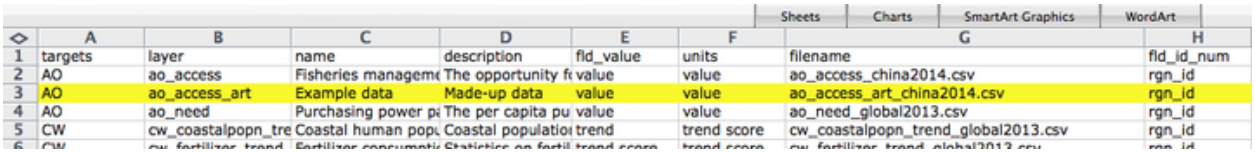
> TIP: In the 'preindex_function' column, you should see what the `year_max`, `status_year`, and `trend_year` say.

### 7.5.3 Example modification:

Suppose your team has decided to add an 'artisanal access' component to the Artisanal Fishing Opportunity goal because of locally available data. Once the data are obtained and properly formatted, the data layer is saved as `ao_access_art`. To include this new information in the goal model, you will need to do the following:

1. register the layer in `layers.csv`
2. update the goal model in `functions.R`
3. update the goal call in `goals.csv`

Step 1. Register in `layers.csv`



| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | targets | layer | name | description | fld_value | units | filename | fld_id_num |
| 2 | AO | ao_access | Fisheries manageme | The opportunity fa | value | value | ao_access_china2014.csv | rgn_id |
| 3 | AO | ao_access_art | Example data | Made-up data | value | value | ao_access_art_china2014.csv | rgn_id |
| 4 | AO | ao_need | Purchasing power pa | The per capita pu | value | value | ao_need_global2013.csv | rgn_id |
| 5 | CW | cw_coastalpopn_tre | Coastal human popu | Coastal populatio | trend | trend score | cw_coastalpopn_trend_global2013.csv | rgn_id |

Figure 43:

Step 2. Update the goal model

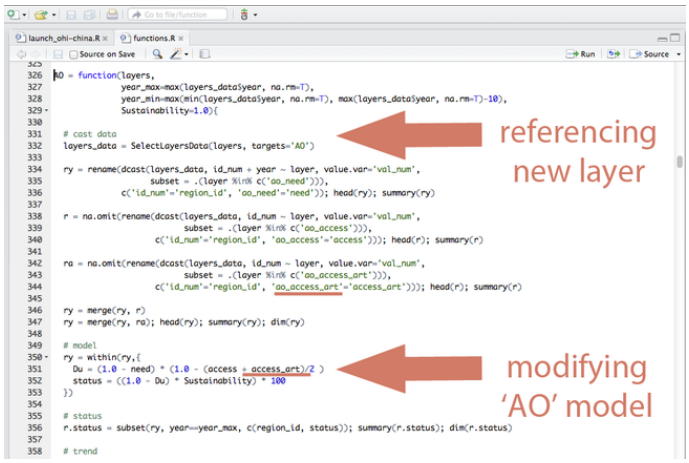



Figure 44:

Step 3. Update goal call in `goals.csv`

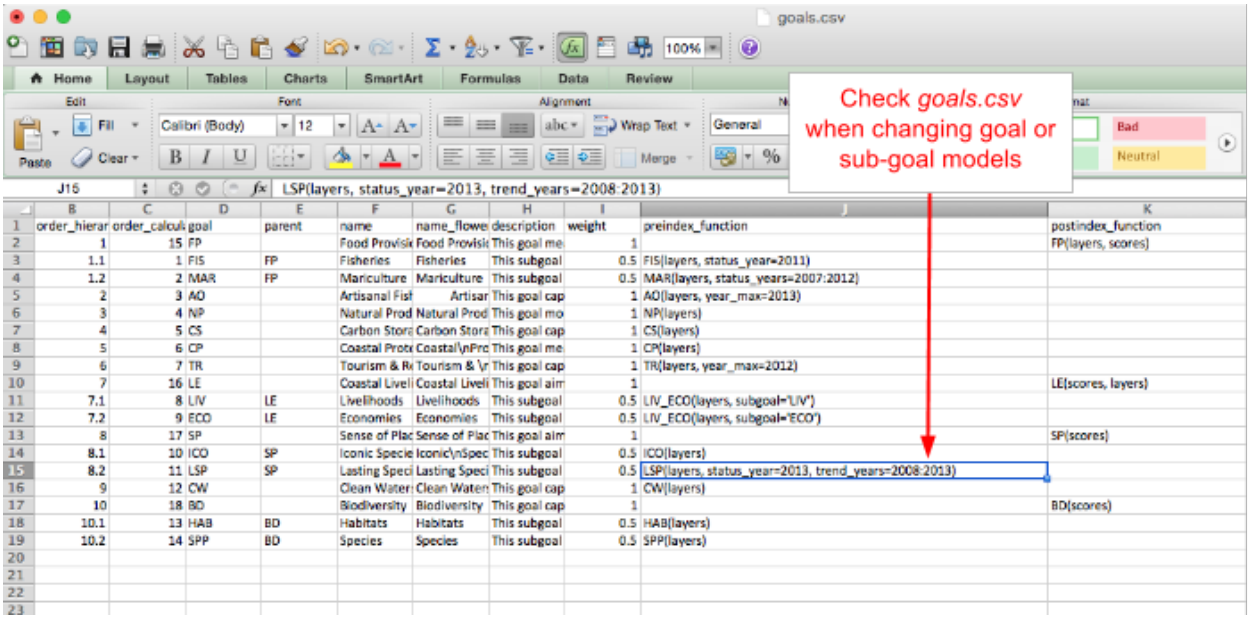



Figure 45: A screenshot of `goals.csv`, used to modify goal model

## 7.6 Removing goals

If a goal is not relevant in your region, it is possible to remove the goal completely from the calculation. There are four places where you will need to remove the reference to this goal. Failing to delete all referenced layers after the goal is deleted will result in errors. To remove goals from your assessment, you will have to do the following:

1. Remove the goal model from `functions.R`
2. Remove the goal's row from `goals.csv`
3. Remove the goal's row from `pressures_matrix.csv`
4. Remove the goal's row from `resilience_matrix.csv`

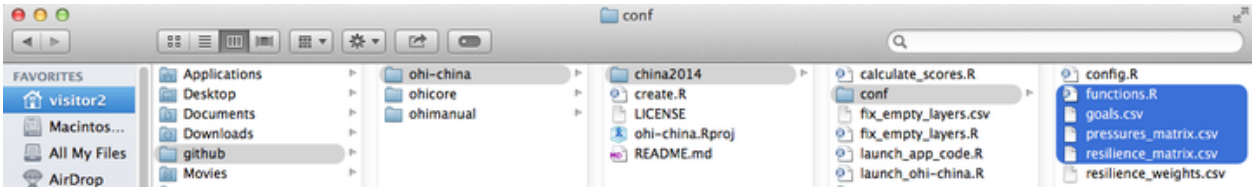



Figure 46:

**Example: Removing carbon storage (CS) goal**

To completely remove the carbon storage goal from Index calculations, you will do the following.

1) Remove the carbon storage (CS) goal model from `functions.R`. Delete the highlighted text in the figure below that references the CS layers and calculates CS goal status, trend, and scores.

2) Remove the CS row from `goals.csv`. Delete the highlighted row in the figure below that contains the CS goal.

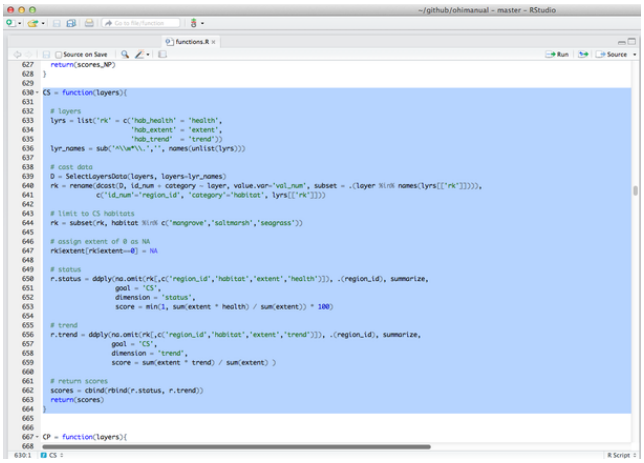


Figure 47:

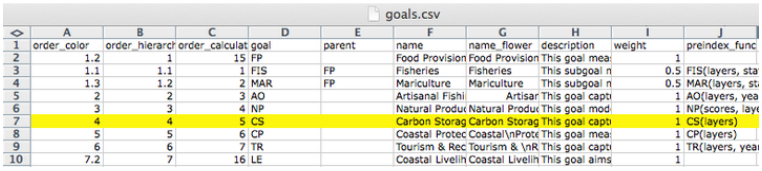


Figure 48:

3) Remove all CS rows from `pressures_matrix.csv`. Delete the highlighted rows in the figure below that contain CS pressures.

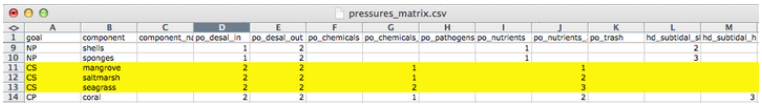


Figure 49:

4) Remove all CS rows from `resilience_matrix.csv`. Delete the highlighted rows in the figure below that contain CS resilience.

## 7.7 Modifying the pressures matrix for goals with categories

### 7.7.1 Background

The pressures and resilience matrix tables identify which pressures and resilience measures (layers) are relevant to which goals and how they are weighted. But pressures and resilience measures can also affect the components within a goal differently. When that is the case, those components can have individual entries (rows) in the pressures and resilience matrix tables and will have pressures and resilience scores calculated individually for each component.

The Toolbox calls these components of a goal 'categories', and knows to calculate pressures and resilience for category elements separately because they are identified in three places: in `pressures_matrix.csv`, `resilience_matrix.csv`, and `config.r`. These files are all located in the `conf` folder. To calculate the

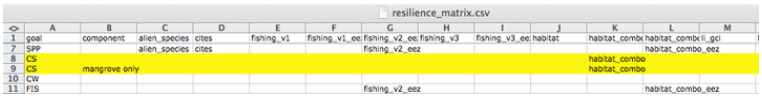



Figure 50:

pressures and resilience scores, the Toolbox uses `config.r` to identify which categories to expect in the matrix tables, and will give a warning if they do not match. `config.r` relies upon the data layers identified in the `pressures_components` and `resilience_components` variables.

In global assessments, there are several goals that have categories indicated in the matrix tables and `config.r` file:

| Goal | Category | layer indicated in `config.r` |
|---|---|---|
| NP | product types | np_harvest_product_weight |
| CS | habitat types | cs_habitat_extent |
| CP | habitat types | cp_habitat_extent_rank |
| HAB | habitat types | hab_presence |
| LIV | industry sectors | le_sector_weight |
| ECO | industry sectors | le_sector_weight |

If you have modified any of the category types in the matrix tables of the above goals, or added new category types to any goals, you will likely need to update the layer indicated in `config.r`. It is also possible to identify individual categories in other goals than those listed above. For example, in the mariculture sub-goal, you could specify the pressures on nearshore mariculture separately from offshore mariculture.

It is important that the file identified in `config.r` does not contain any NA values.

### 7.7.2 Example 1: Pressures

Here is an example of how to modify existing category types for the natural products goal.

In the China OHI+ assessment there are three natural product types (seasalt, sea chemicals, and sea medicine), which differ from those assessed in the global assessments (corals, fish_oil, ornamentals, seaweeds, sponges). After modifying and registering the appropriate data layers and updating the NP function in `functions.r`, it is time to update the natural product types in `pressures_matrix.csv`, `resilience_matrix.csv`, and `config.r`.

- to update `pressures_matrix.csv` and `resilience_matrix.csv`, make sure that each product type has a separate row, with the appropriate pressures identified and weights attributed.
- to update `config.r`, check that the data layer identified in the `pressures_components` and `resilience_components` has the same category types.

When you run `calculate_scores.r`, the following warning will alert you that there is a mis-match between category types identified in the matrix and `config.r`:

```
Calculating Pressures...
The following components for NP are not in the
aggregation layer np_harvest_product_weight categories
(corals, fish_oil, ornamentals, seaweeds, sponges):
seasalt, sea_chemicals, sea_medicine
```

This message indicates that the `np_harvest_product_weight` layer identifies five categories (corals, fish_oil, ornamentals, seaweeds, sponges) but the `pressures_matrix.csv` indicates three (seasalt, sea_chemicals, sea_medicine).

To ensure that pressures are calculated correctly for the categories in your assessment, you will need to change the layer identified in `config.r`.

Note that more subtle examples of these mismatch between the categories identified in `pressures_matrix.csv` and `config.r` can also occur. For example, after updating the carbon storage layers and goal model in the China OHI+ assessment, the following warning message appeared when running `calculate_scores.r`:

```
Calculating Pressures...
The following components for CS are not in the aggregation layer
cs_extent categories (saltmarshes, seagrasses, mangroves):
mangrove, saltmarsh, seagrass
```

The problem here is that the categories identified in `config.r` (saltmarshes, seagrasses, mangroves) are plural, whereas the categories identified in the pressures matrix (mangrove, saltmarsh, seagrass) are singular, and the Toolbox needs exact matches. To fix this warning, you need to update the pressures matrix with the plural names.

### 7.7.3 Example 2: Resilience

For resilience calculations, the proper categories also need to be identified both in `resilience_matrix.csv` and `config.r`. If there is a mismatch, you will see the following message:

```
Calculating Resilience...
Note: each goal in resilience_matrix.csv
must have at least one resilience field
Based on the following components for NP:
corals
fish_oil
ornamentals
seaweeds
shells
sponges
```

With resilience, if we update only the `resilience_matrix.csv` but not `config.r`, we get the following error message instead of the warning message we saw for pressures above.

```
Based on the following components for NP:
  seasalt
  sea_chemicals
  sea_medicine
Error in subset.default(SelectLayersData(layers, layers = lyrs),
id_num ==  : object 'id_num' not found
In addition: Warning messages:
1: Grouping rowwise data frame strips rowwise nature
2: In left_join_impl(x, y, by$x, by$y) :
  joining factors with different levels, coercing to character vector
```

This error can be fixed by updating `config.r` with a layer identifying the appropriate categories.

## 7.8 Other example modifications

### 7.8.1 Preparing the fisheries sub-goal

Here is some background information about how to prepare fisheries data layers for the Toolbox.

**Data layers used by the Toolbox:**

- `fis_b_bmsy`
- `fis_meancatch`
- `fis_proparea_saup2rgn`
- `fp_wildcaught_weight`

#### 7.8.1.1 Description of data layers `fis_b_bmsy`

- *for species*: B/Bmsy estimate (either from formal stock assessment, or from a data-poor method such as CMSY)
- *for genus/family/broader taxa*: the toolbox will use median B/Bmsy from species in that region + a penalty for not reporting at species level. In order for the code to assign the correct penalty, the taxa need to include a numerical code of 6 digits, where the first digit behaves like an ISSCAAP code (the standardized species codes used by FAO): 6 means species, 5 means genus, 4 to 1 are increasingly broad taxonomic groups
- *data source (for CMSY)*: catch time-series (at least 10 years of catch >0), species resilience (if available)

**Example data:**

| fao_id | taxon_name | year | b_bmsy |
|---|---|---|---|
| 51 | Ablennes hians | 1985 | 1.112412 |
| 51 | Ablennes hians | 1986 | 1.222996 |
| 51 | Ablennes hians | 1987 | 1.371058 |

NOTE: if a species that is caught in different sub-regions belongs to the same population, you don't want to split the catch among sub-regions, instead, you want to sum catch across all sub-regions, so you can calculate B/Bmsy for the whole population. For the global analysis we grouped all species catch by FAO major fishing area (www.fao.org/fishery/area/search/en), indicated in the column *fao_id*, assuming that all species caught within the same FAO area belonged to the same stock, while we assumed that the same species, if caught in a different fishing area, belonged to a separate stock.
Use *fao_id* as an identifier that separates different fisheries 'stocks' belonging to the same species. If you don't have multiple stocks in your study area, set all *fao_id* = 1.

`fis_meancatch`:

- average catch across all years, per species, per region

- *data source*: catch time-series (at least 10 years of catch >0), with a unique identifier for each population that you want to assess separately

**Example data:**

| fao_saup_id | taxon_name_key | year | mean_catch |
|---|---|---|---|
| 37_8 | Aristeus antennatus_690051 | 2014 | 14.24398116 |
| 37_8 | Atherinidae_400218 | 2014 | 27.30120156 |
| 37_8 | Balistes capriscus_607327 | 2014 | 3.247883895 |

The *taxon_name_key* column indicates the name of the species (e.g. Aristeus antennatus) and its 'taxonkey'. The taxonkey is a 6 digit numeric code used by the Sea Around Us Project, modified from FAO codes. The important element of this code is the first digit, because it reflects the taxonomic level (6=species, 5=genus,

4=family, etc.) of the reported catch.The toolbox uses this first digit to assign a score to all catch that was not reported at species level, taking the median of the B/Bmsy of assessed species, and adding a penalty that is increasingly strong for coarser taxa.

`fis_proparea_saup2rgn`:

- a conversion file that, for each region for which catch is reported, tells us what proportion of that region falls within each of the final OHI reporting regions.

**Example data:**

| saup_id | rgn_id | prop_area |
|---|---|---|
| 166 | 1 | 1.0 |
| 162 | 2 | 1.0 |
| 574 | 3 | 0.7 |
| 37 | 4 | 0.8 |

**Specific instances:**

*only if catch is reported for different regions than the ones used for the OHI assessment:* this should be calculated using spatial analyses of overlap of the spatial units at which catch is reported with the spatial units at which the OHI assessment will be reported. The global data was reported by subregions (*saup_id*) and in some cases multiple subregions were part of the same, larger EEZ. Since for OHI we wanted results by EEZ (*rgn_id*), in those cases we needed to combine results from the subregions to get the final score, based on their size relative to the total EEZ size (*prop_area*). *If catch is reported for the same areas for which OHI is calculated:* then all the *prop_area* are = 1. *If catch is reported for the whole area of the assessment, but you want to calculate a separate OHI score for different sub-regions:* for each OHI reporting region (*rgn_id*) you'll repeat the same region in the *saup_id* column, and *prop_area* will be =1. This effectively means all the reporting regions will get assigned 100% of the catch and will have the same final stastus and trend score for the fisheries goal (but may have different pressures and resilience scores, if those layers are different in each sub-region).

`fp_wildcaught_weight`:

*only needed if there is mariculture*: for each region, this represents the relative proportion of catch coming from wild caught fisheries versus mariculture. The layer is used to weight how much the fisheries score influences the final food provision score, the higher the fisheries catch, the more the food provision score will reflect the fisheries score, and vice-versa if mariculture has a higher catch. (NOTE that, before all mariculture harvest from all species gets summed, the mariculture harvest for each species is smoothed and then multiplied by the resilience score).

#### 7.8.1.2 Running CMSY model Sample data to run CMSY:

| id | stock_id | res | ct | yr |
|---|---|---|---|---|
| 6 | Acanthistius brasilianus_41 | Medium | 100 | 1950 |
| 23 | Acanthurus dussumieri_61 | | 0.059250269 | 1950 |
| 24 | Acanthurus dussumieri_71 | | 0.190749971 | 1950 |
| 25 | Acanthurus lineatus_61 | Low | 12.74821966 | 1950 |

The current CMSY script produces an output that looks something like this (split into 2 tables):

| stock_id | convergence | effective_sample_size | yr | b_bmsy | b_bmsyUpper |
|---|---|---|---|---|---|
| Ablennes hians_51 | SC | 30974 | 1985 | 1.112412 | 1.8 |
| Ablennes hians_51 | SC | 30974 | 1986 | 1.222996 | 1.768895 |

| stock_id | yr | b_bmsyLower | b_bmsyiq25 | b_bmsyiq75 | b_bmsyGM | b_bmsyMed |
|---|---|---|---|---|---|---|
| Ablennes hians_51 | 1985 | 1 | 1 | 1 | 1.093932 | 1 |
| Ablennes hians_51 | 1986 | 1.014688 | 1.075699 | 1.298437 | 1.209005 | 1.160329 |

where *stock_id* is the unique identifier for each stock that was used in the input file, *convergence* indicates whether the model converged and how strongly ('SC' = strong convergence), *effective_sample_size* reports the number of iterations used, *yr* = year, b_bmsy = B/Bmsy for the corresponding year (based on the median of all the estimated values: reccomended), b_bmsyUpper = B/Bmsy at the upper 95% bootstrapped confidence bound, b_bmsyLower = B/Bmsy at the lower 95% bootstrapped confidence bound, b_bmsyiq25 = B/Bmsy at the first quartile, b_bmsyiq75 = B/Bmsy at the third quartile, b_bmsyGM = B/Bmsy based on the geometric mean of estimates, b_bmsyMed = B/Bmsy based on the median of estimates.

**How to:**

**1. Include resilience in the CMSY code:**

In the CMSY R script, in the PARAMETERS section, replace the following:

```
start_r    <- c(0.01,10)  ## disable this line if you use resilience
with

  if(res == "Very low"){
    start_r  <- c(0.015, 0.1)
  } else {
    if(res == "Low"){
      start_r  <- c(0.05,0.5)
    } else {
      if(res == "High"){
        start_r  <- c(0.6,1.5)
      } else {
        start_r  <- c(0.1,1)
      }
    }
  }
```

Figure 51:

**2. Make assumptions about fisheries regulations:**

If you assume that fisheries are depleted and there isn't very much fisheries regulation, and you are using the CMSY method to assess B/Bmsy, the original model may work well. If, however, the catch of a species declined because fisheries regulations have closed or limited the fishery, or if a fishery was abandoned for economic reasons (e.g., change in consumer prefereces, market price dynamics, etc.), the model may be too pessimistic and understimate B/Bmsy. In that case it may be best to use a version with a uniform prior on final biomass, instead of the constrained prior.
The original constrained prior on final biomass is set by this line within the code:

```
finalbio    <- if(ct[nyr]/max(ct) > 0.5) {c(0.3,0.7)} else {c(0.01,0.4)}
```

The model uses a uniform prior if that line is replaced with:

```
finalbio    <- c(0.01,0.7)
```

**3. Use data at a different spatial resolution than the final assessment:**

See notes above for `fis_proparea_saup2rgn`

**4. Calculate B, or Bmsy:**

The CMSY model calculates B/Bmsy as a ratio, it does not estimate the two variables separately.

**5. Use catch per unit of effort (CPUE):**

The CMSY model requires total biomass removed by fisheries, and uses catch as a proxy for that. It cannot use CPUE. Other more sophisticated stock assessment models use CPUE and may be employed. We do not provide documentation for the use of these other models.

**6. Use other life-history characteristics, in addition to resilience:**

The CMSY model does not use more detailed information. Other more sophisticated stock assessment models use other life-history traits such as fecundity, larval dispersal, r, K, Lmax, etc., and may be employed. We do not provide documentation for the use of these other models.

**7. Create a 'taxonkey' to assign to each species:**

When replacing the SAUP_FAO data with your own data, assign a key of 600000 to all species. For all catch that is reported at genus or coarser taxonomic level, you will have to choose an appropriate taxonkey. You can create your own key, from 100000 to 500000, based on your own judgment of how many species may be reported under that same denomination, and how different they may be (all that matters for the toolbox code is whether the number starts with a 1,2,3,4,5 or 6 with 1 being the coarsest, such as 'miscellaneous marine animals', or 'crustaceans nei').

#### 7.8.1.3 Resources

Martell, S & Froese, R (2013) "A simple method for estimating MSY from catch and resilience". *Fish and Fisheries*, DOI: 10.1111/j.1467-2979.2012.00485.x. Downloadable here

Rosenberg, A.A., Fogarty, M.J., Cooper, A.B., Dickey-Collas, M., Fulton, E.A., Gutiérrez, N.L., Hyde, K.J.W., Kleisner, K.M., Kristiansen, T., Longo, C., Minte-Vera, C., Minto, C., Mosqueira, I., Chato Osio, G., Ovando, D., Selig, E.R., Thorson, J.T. & Ye, Y. (2014) Developing new approaches to global stock status assessment and fishery production potential of the seas. *FAO Fisheries and Aquaculture Circular No. 1086.* Rome, FAO. 175 pp. Downloadable here

## 7.9 Updating the WebApp's pages

The WebApp displays input layers on several pages: on the App page, Layers page, and Scores page. These input layers are displayed from `layers.csv` and the layers within the `layers` folder and the scores are displayed from `scores.csv`. While the input layers and scores will be automatically displayed on the WebApp, there is other content on WebApps pages that can be edited by your team and displayed. You will likely spend the most time updating the equations displayed on the Goals page to be consistent with the updated methods you have used in your assessment.

For the WebApp to display the pages properly, not everything on each page can be edited as it is written in a language to create the website. But it is possible to explore the files and update much of the text that is displayed while maintaining the required formatting. To do this, follow the instructions in `copy_webapps_templates.r`, which creates a folder called `webapps_templates` and copies template files there.

It is best to edit the files in RStudio: you will be able to view your work as it will be displayed on the WebApp by clicking the 'Preview HTML'.

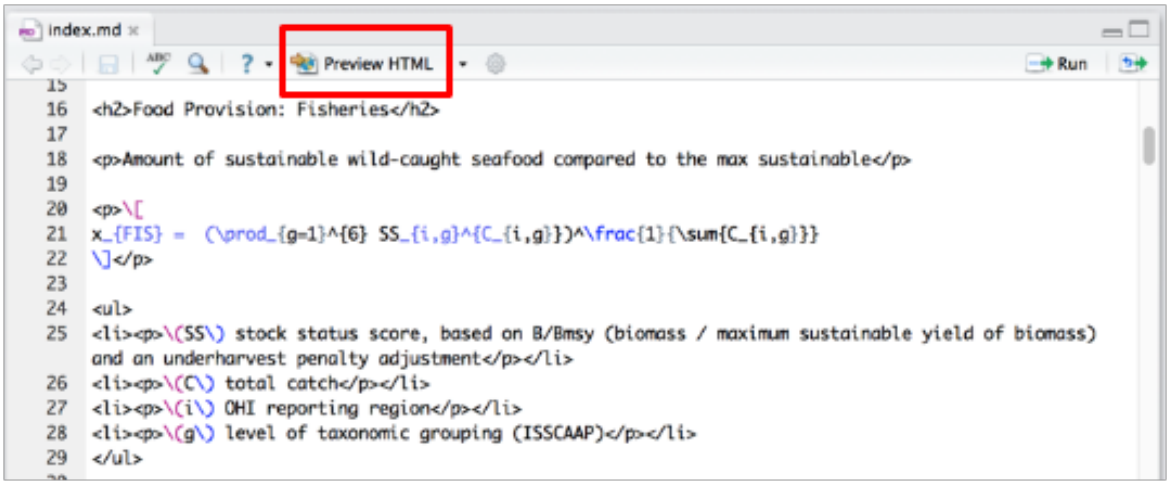



Figure 52:

### 7.9.1 Regions

You may have redefined the spatial boundaries of the regions used in your assessment, or you may want to update the information provided about them. This can be done with the file called `webapps_templates/regions.brew.md`.

### 7.9.2 Layers

Most of the information displayed on the Layers page of the WebAppis taken from the `layers.csv`, and therefore to modify any information about specific data layers, you will need to modify the `layers.csv` file within the draft branch of your repository. However, you are able to edit the header text information at the top of the Layers page if you wish. This can be done with the file called `webapps_templates/layers.brew.md`.

### 7.9.3 Goals

You will likely spend the most time modifying the information displayed on the Goals page, as these show and describe the models used in the assessment. Text can be modifed with the file called `webapps_templates/goals.brew.md`.

To edit the goal equations themselves, you will edit the `goals.Rmd` found in the `conf` folder (example: `ecu/subcountry2014/conf/goals.Rmd`. This is an Rmarkdown file, with equations written in LaTex. When rendered by RStudio or the WebApp, it displays nicely formatted. To update model equations, you will need to use the LaTex format. You can learn the syntax by studying how the equations from the global assessments are displayed, and from many resources online. One resource is https://en.wikibooks.org/wiki/LaTeX/Mathematics. Learn more about .Rmd formatting at http://shiny.rstudio.com/articles/rmarkdown.html.

### 7.9.4 Scores

The scores displayed on the Scores page of the WebApp are the calculated scores from the `scores.csv` file in the draft branch, and therefore cannot be modified. However, you are able to edit the header text information at the

top of the Scores page if you wish. This can be done with the file called `webapps_templates/scores.brew.md`.

## 7.10 R Tutorials for OHI

Ocean Health Index R code uses several packages and best practices to faciliate understanding and collaboration. These approaches are presented here, along with examples using data included in global OHI assessments.

This document describes several packages that are used extensively in OHI assessments and introduces you to typical coding practices commonly seen in OHI scripts and functions.

Also see the accompanying R script to test examples using these packages.

### 7.10.1 R Very Basics:

- Have you already downloaded and installed R?
- Have you already downloaded and installed RStudio?
- Have you walked through the excellent interactive tutorials from swirl?

### 7.10.2 `tidyr` functions

'Tidy' up your messy data using `tidyr` to make it easier to work with. The 'tidy tools' functions in the `dplyr` package work best with tidy data.

From Hadley Wickham's *Tidy Data* paper: >It is often said that 80% of data analysis is spent on the cleaning and preparing data. And it's not just a first step, but it must be repeated many over the course of analysis as new problems come to light or new data is collected. To get a handle on the problem, this paper focuses on a small, but important, aspect of data cleaning that I call data tidying: structuring datasets to facilitate analysis.

From RStudio's introduction to `tidyr`:

> The two most important properties of tidy data are: 1. Each column is a variable. 2. Each row is an observation.
>
> Arranging your data in this way makes it easier to work with because you have a consistent way of referring to variables (as column names) and observations (as row indices). When you use tidy data and tidy tools, you spend less time worrying about how to feed the output from one function into the input of another, and more time answering your questions about the data.

`gather()` is arguably the most useful function in `tidyr`, and is explained in more detail below. `spread()` and `separate()` are other useful functions in `tidyr`.

Other 'tidyr' references: * Hadley Wickham's *Tidy Data* paper: Download the pre-print version for the whys and hows of tidy data. * Cran tidy data vignette: An informal and code heavy version of Hadley's full *Tidy Data* paper. * RStudio Blogs: Introducing tidyr: Basics and philosophy of `tidyr` * swirl tutorial package: A tutorial package built directly into R. Section 2: 'Getting and Cleaning Data' runs you through `dplyr` and `tidyr` basics * R data wrangling cheat sheet: A quick reference guide to `tidyr` and `dplyr` functions

#### 7.10.2.1 tidyr::gather() Description

`gather()` takes data organized in rows and collapses them into a column format (a key column and a value column), duplicating all other columns as needed. Use `gather()` when your data is organized in "wide" format, in which some of your variables are in row form, rather than column form. Another `tidyr` function, `spread()`, is more or less the reverse of `gather()`, to reformat long data into wide data. It is more difficult to work with wide data, but may be more convenient for examining data in a table format.

Note: `gather()` essentially replaces `melt()` in `plyr` package.

**Example**

The sample data set (see intro) contains harvest data of a number of marine commodities, separated by country, commodity, and year. In its original form, the harvest data (in tonnes) is spread across five different harvest years. * Counter to 'tidy data' principles, we have multiple columns (X2007:X2011) representing a single variable (year), and multiple observations of harvest tonnage in each row. * To transform this into 'tidy data' we will gather the five annual harvests into a single column called 'tonnes' and note the year of harvest in a new column called 'year'.

The example in the figure below shows how the original wide data is transformed into long data using the command `gather`. Here are two ways of acheiving this:

1. Here, information from columns X2007 through X2011 are gathered into a single column called `year`, and the information in each column are put into a new column called `tonnes`.

```
data_long <- data_wide %>% gather(year, tonnes, X2007:X2011)
```

2. Here, the `-` unselects the named columns, so they will not be gathered; all other columns are gathered into columns named `year` and `tonnes`. This approach will yield the same result.

```
data_long <- data_wide %>% gather(year, tonnes, -Country, -Commodity, -Trade)
```

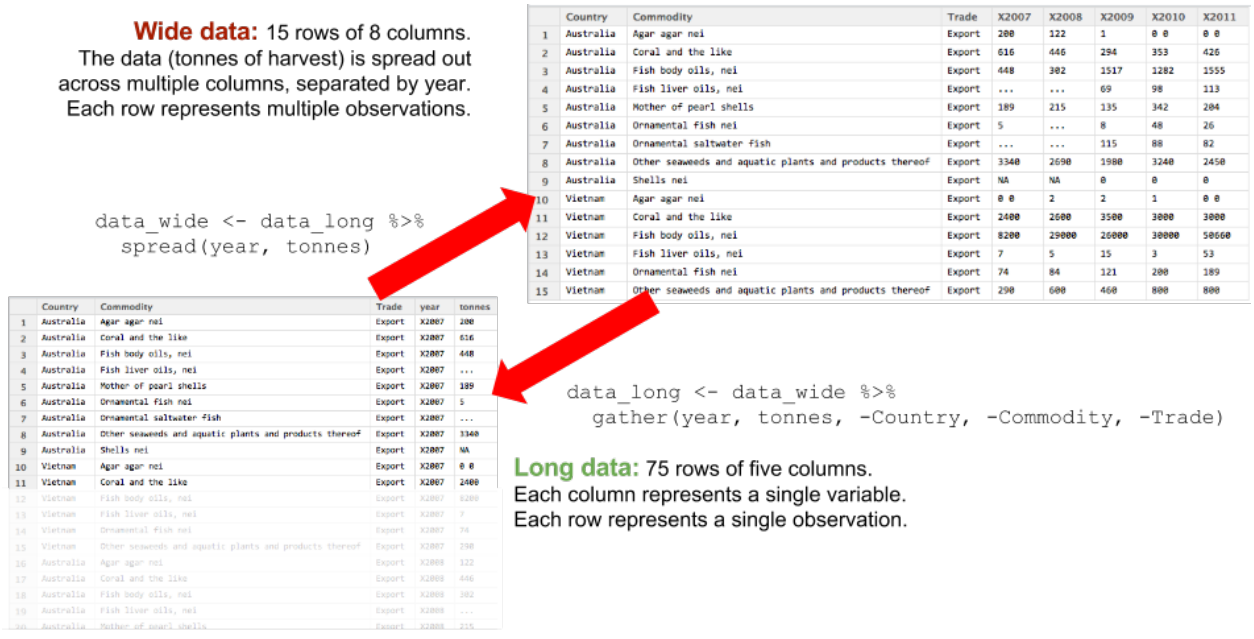



Figure 53: wide data to long data using gather() and spread()

### 7.10.3 `dplyr` functions

The `dplyr` package includes a number of functions to easily, quickly, and intuitively wrangle your data. Here is a quick introduction with examples from data used in the Ocean Health Index.

From RStudio's introduction to `dplyr`:

> The bottleneck in most data analyses is the time it takes for you to figure out what to do with your data, and `dplyr` makes this easier by having individual functions that correspond to the most common operations...
>
> Each function does one only thing, but does it well.

The most important `dplyr` functions to understand for data processing will be `group_by()`, `mutate()`, and `summarize()`. Also important, `dplyr` introduces the ability to perform subsequent functions in a logical and intuitive manner, using the `%>%` chain operator.

- `%>%` (chaining operator): allows sequential chaining of functions for cleaner, easier-to-read code
- `dplyr::select()`: selects variables to be retained or dropped from dataset
- `dplyr::filter()`: filters data set by specified criteria
- `dplyr::arrange()`: sorts dataset by specified variables
- `dplyr::mutate()`: adds variables or modifies existing variables
- `dplyr::summarize()`: uses analysis functions (sum, mean, etc) to summarize/aggregate specified variables
- `dplyr::group_by()`: groups data by specified variables, allowing for group-level data processing.

Other `dplyr` references:

- RStudio blogs: Introducing dplyr:: philosophy, examples, and basics of `dplyr`
- Cran dplyr vignette: Walkthrough of `dplyr` with examples
- `dplyr` and pipes: the basics: More examples of `dplyr` functions, and more depth on `%>%`
- swirl tutorial package: A tutorial package built directly into R. Section 2: 'Getting and Cleaning Data' runs you through `dplyr` and `tidyr` basics
- R data wrangling cheat sheet: a quick reference guide to `tidyr` and `dplyr` functions

#### 7.10.3.1 %>% operator Description

The `%>%` operator allows you to 'pipe' or 'chain' a number of function calls, in which the output dataframe of one function is fed directly into the next function as the input dataframe. This lets you avoid creating temporary variables to store intermediate values, and lets you avoid nesting multiple functions. Using `%>%` makes your code more elegant, streamlined, and easy to read since you are able to write your code on multiple indented lines. From `dplyr` and pipes: the basics:

> OK, here's where it gets cool. We can chain `dplyr` functions in succession. This lets us write data manipulation steps in the order we think of them and avoid creating temporary variables in the middle to capture the output. This works because the output from every `dplyr` function is a data frame and the first argument of every `dplyr` function is a data frame.

**Usage**

```
data_out <- f(data_in, args)
  # standard function call

data_out <- data_in %>% f(args)
  # function call using %>% operator. data_in is passed as first argument
  # of function().

data_out <- data_in %>%
  f1(args1) %>%
  f2(args2) %>%
  f3(args3) %>% ...
  # Output of function can be passed to another function immediately,
  # without need for temporary storage. Indented format for legibility,
  # see how pretty it looks?
```

**Example**

```
### Bad!  Nested functions: read from inside out - hard to decipher
  h_recent_totals1 <- arrange(mutate(filter(group_by(harvest, country, commodity),
    year >= 2009), harvest_tot = sum(tonnes, na.rm = TRUE)), country, commodity)

### Better: Line by line. Easier to read, but have to wait for the end to see
### what it does.  Temp variables add more places for errors and bugs.
  h_temp <- group_by(harvest, country, commodity)
  h_temp <- filter(h_temp, year >= 2009)
  h_temp <- mutate(h_temp, harvest_tot = sum(tonnes, na.rm = TRUE))
  h_recent_totals2 <- arrange(h_temp, country, commodity)

### Best!  Chained format intuitively links together the functions. Saves
### typing, fewer opportunities for errors, easier to debug. The %>% operator
### automatically indents each following line for easy reading.
  h_recent_totals3 <- harvest %>%
    group_by(country, commodity) %>%
    filter(year >= 2009) %>%
    mutate(harvest_tot = sum(tonnes, na.rm = TRUE)) %>%
    arrange(country, commodity)
```

#### 7.10.3.2 dplyr::select() Description

`select()` allows you to choose specific columns/variables from your dataset, and drop all others. Alternately, you can select specific variables to drop, leaving others in place. `rename()` is a relative of `select()` that allows you to rename variables, while leaving all variables in place.

**Examples**

The sample dataset includes the annual harvest, in tonnes, of a number of commodities exported by two countries. Type of trade provides no information (it is all Export), so that variable can be dropped. The names of all the variables should be converted to lower-case, to match the OHI style guide. See the figure below.

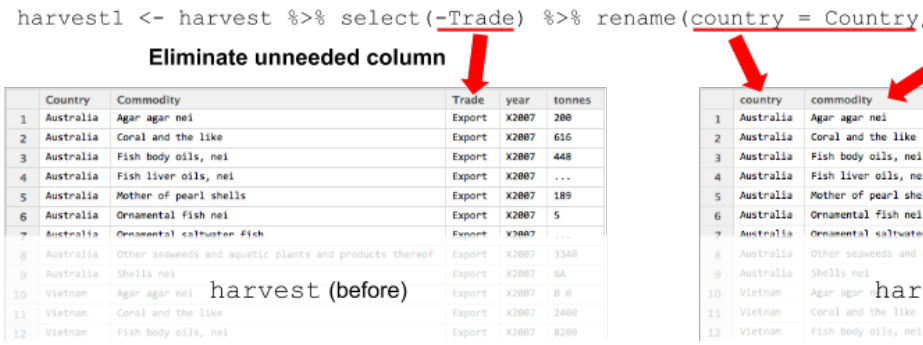
```
### Example 1:
harvest <- harvest %>%
  select(Country, Commodity, year, tonnes)
  ### Selects the named variables, and drops all others.  Useful to choose a
```

```
  ### subset of key variables from a complicated data set.

### Example 2 (same result as example 1):
harvest <- harvest %>%
  select(-Trade)
  ### Using the '-' drops 'Trade' column and leaves other variables intact.
  ### Useful if you would like to clear out temporary variables.

harvest <- harvest %>%
  rename(country = Country, commodity = Commodity)
  ### Drops no variables. Syntax: rename(new_var_name = old_var_name) w/o quotes.
```

Using the chain operator, we can string these two functions into one smooth, easy-to-read flow:

```
harvest1 <- harvest %>%
  select(-Trade) %>%
  rename(country = Country, commodity = Commodity)
```

The `harvest` data is fed into `select()`, and the output is fed into `rename()`. The final output of this complete flow is assigned to the new variable `harvest1`.




#### 7.10.3.3 dplyr::filter() Description

`filter()` allows you to select observations (rows) that match search criteria, using values in specified variables (columns). Drops all observations that do not match the criteria. * Use logical operators & and | to filter on multiple criteria simultaneously

**Example**

```
harvest_vnm  <- harvest %>%
  filter(country == 'Vietnam')
  ### Single criterion filter: keeps only data with country matching 'Vietnam'.

h_vnm_recent <- harvest %>%
  filter(country == 'Vietnam' & year >= 2009)
  ### filter with multiple criteria: selects 'Vietnam' data from 2009 or later.
```

#### 7.10.3.4 dplyr::arrange() Description

`arrange()` sorts observations (rows) based upon a specified variable or list of variables. Does not actually change the data in any way, only the appearance. Useful for inspecting your data after each processing step.

**Example**

```
harvest_sorted <- harvest %>%
  arrange(country, commodity, year)
  ### Sorts commodity harvest values for each country, chronologically

harvest_sorted <- harvest %>%
  arrange(country, commodity, desc(year))
  ### Sorts harvest values by most recent year (descending order)
```

#### 7.10.3.5 dplyr::mutate() Description

`mutate()` is a powerful and useful tool for processing data. You can add new variables or modify existing variables, using all variety of functions to perform operations on the dataset. `mutate()` works well with `group_by()` to perform calculations and analysis at a group level rather than dataset level.

**Example**

From the sample data set (see figure below), we would like to:

- Remove the 'X' from the 'year' values.
- Translate the text codes in 'tonnes' into numbers and NAs. These codes are specific to FAO's data reporting format: `...` is the same as `NA`, and `0 0` means greater than zero, but less than half a tonne.
- Convert these text fields into numeric fields so they can be analyzed properly.

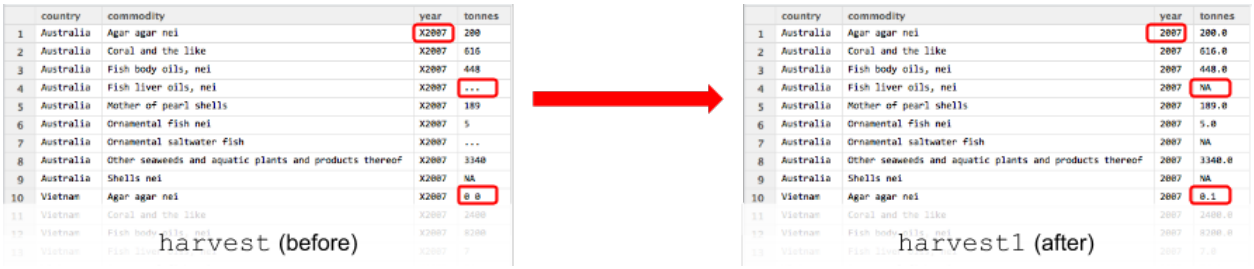



Figure 54: Using mutate to alter data in a dataframe

```
library(stringr)   ### to access 'str_replace()' string functions

harvest1 <- harvest %>%
  mutate(
    year   = str_replace(year,   fixed('X'),    ''),  # remove the 'X'
    tonnes = str_replace(tonnes, fixed('...'),  NA),  # replace '...' with 'NA'
    tonnes = str_replace(tonnes, fixed('0 0'), 0.1),  # replace '0 0' with '0.1'
    tonnes = ifelse(tonnes ==  '', NA, tonnes)) %>%
  mutate(
    tonnes = as.numeric(as.character(tonnes)),
    year   = as.integer(as.character(year)))
```

Notes: * In this example, no new variables were added. Multiple variables can be changed with one call to `mutate()`. Multiple modifications to 'tonnes' happen sequentially, so order is important. * The `as.numeric(as.character(...))` gets around the fact that these text variables are stored as 'factor' class, rather than 'character' class. `as.character()` forces them into character class, and then the `as.numeric()` can convert the character strings to numeric where applicable. Similar for `as.integer(...)`

#### 7.10.3.6 dplyr::summarize() ( or summarise() ) Description

`summarize()` combines multiple values of a variable into a single summary value. `summarize()` works well with `group_by()` - for grouped data, each group will be summarized and reported separately. For ungrouped data, the summary covers the entire dataset.

- `summarize()` compresses the dataset and drops individual observations. To maintain individual observations, consider creating a summary variable using `mutate()` instead.
- `NA` values can be problematic - use `na.rm=TRUE` or similar methods.

**Example**

To determine the total harvest of each country, for each commodity:

```
h_summary <- harvest %>%
  group_by(country, commodity) %>%
  summarize(harvest_tot = sum(tonnes, na.rm = TRUE)) %>%
  ungroup()
```

#### 7.10.3.7 dplyr::group_by() Description

`group_by()` allows you to easily group a dataset by one or more variables/columns.
By itself, it does nothing to change your data. But once your dataset has been sorted into useful groups, other `dplyr` functions will operate on each group separately, rather than operating on the entire dataset. * The function `groups(data)` reports back the current grouping status of dataframe `data`.

- `group_by()` alters the grouping, but does not alter the sort order.
  `arrange()` does not alter the current grouping - it will sort by groups first, then sorts within each group.
- Multiple calls to `group_by()` will reset the groupings each time (by default), rather than adding additional layers of groups.
- Once you have finished with your operation at the group level, it is a good practice to use the `ungroup()` function to remove the groupings, to avoid unintended consequences due to forgotten `group_by()` calls.

**Example**

If you want to find the total tonnage harvested for each commodity for each country, you would want to group by country and by commodity, and then perform a `sum()` function on the grouped data. Two options presented here: `summarize()` to collapse data to just the summary, and `mutate()` to add a new column that includes the summary values.

```
h_tot_sum <- harvest %>%
  group_by(country, commodity) %>%
  summarize(harvest_tot = sum(tonnes, na.rm = TRUE))
    ### Summarize information by collapsing each group to a single summary value
    ### (total tonnage by commodity by country). Note ungroup() at end.

h_tot_mut <- harvest %>%
  group_by(country, commodity) %>%
  mutate(harvest_tot = sum(tonnes, na.rm = TRUE)) %>%
  arrange(country, commodity) %>%
  ungroup()
    ### Summarize information by creating a new variable to contain summary
    ### value; report value for every observation. Note ungroup() at end.
```

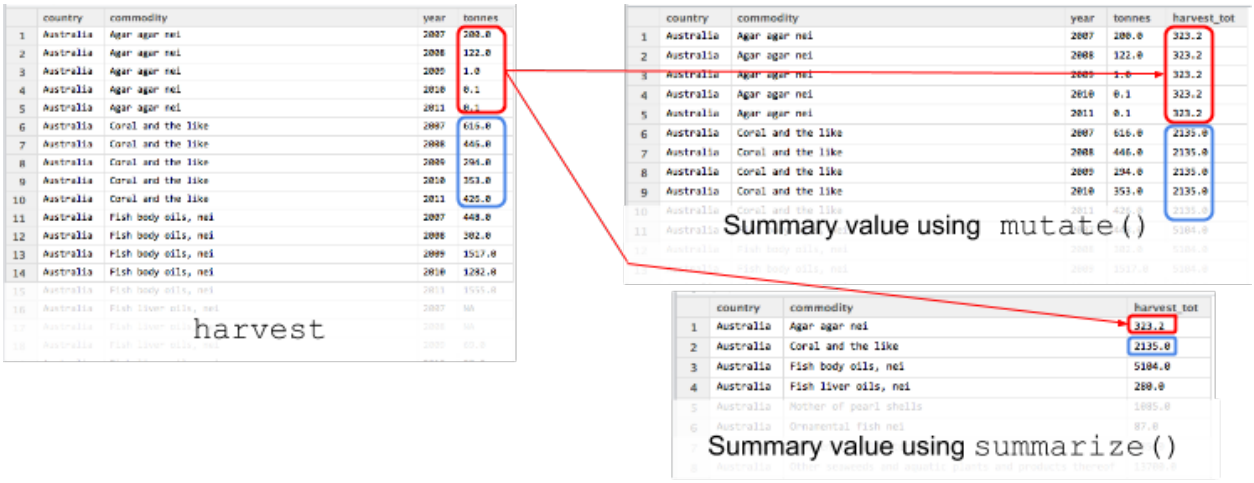



Figure 55: group_by to find group-level information

### 7.10.4 Coding style

> Code unto others as you would have them code unto you.

Why style? ask Hadley Wickham, developer of many wonderful R packages:

> Good style is important because while your code only has one author, it'll usually have multiple readers. This is especially true when you're writing code with others. In that case, it's a good idea to agree on a common style up-front. Since no style is strictly better than another, working with others may mean that you'll need to sacrifice some preferred aspects of your style.

The Ocean Health Index is founded upon principles of open-source science, so our code should be not just available, but legible to others. For OHI+, we expect people to modify code to implement new goal models, and we may need to provide support in developing and debugging their code.

Certain coding techniques are more efficient than others (e.g. in R, looping across elements in a vector is much slower than operating on the entire vector at once), but rarely does OHI code push any performance envelopes. Much more of our time is spent writing code, translating old code into new models, and debugging. Transparent, readable code will save more time in the future than a perfectly-optimized but opaque algorithm.

Readable code is:

- collaborative
- easier for others to understand and debug
- easier for others to update and modify
- easier for 'future you' to interpret what 'past you' meant when you wrote that chunk of code.

Check out Hadley Wickham's style guide:

- How many of these suggestions are second-nature to you? how many are you guilty of breaking?
- Note that these are guidelines, not rules; non-stylish code can still work.

#### 7.10.4.1 Best practices for coding in OHI assessments:

- use a consistent format for variable names, filenames, function names, etc.
  - `lower_case_with_underscores` (preferred) or `camelCase` (ok I suppose)
    - not `periods.in.between`
  - use names that are brief but intuitive
- Comment clearly for your own purposes, and for others.
  - Comment on the purpose of each important block of code.
  - Comment on the reasoning behind any unusual lines of code, for example an odd function call that gets around a problem.
- Take advantage of R Studio section labels functionality:
  - If a comment line ends with four or more -, =, or # signs, R Studio recognizes it as a new section.
  - Text within the comment becomes the section name, accessible in the drop-down menu in the bottom left of the RStudio script window.
- use <- to assign values to variables (not necessary, but preferred)
- use %>% to create intuitive chains of related functions
  - one function per line
  - break long function calls into separate lines (e.g. multiple mutated variables)
- use proper spacing and formatting for legibility
  - don't crowd the code - use spaces between math operators and after commas
  - use indents to indicate nested or sequential/chained code
  - break sequences or long function calls into separate lines logically - e.g. one function call per line
- use functions to add intuitive names to chunks of code
- Use 'tidy data' practices - take advantage of `tidyr`, `dplyr`
  - clean up unused columns using `select(-colname)`
- if you are working on an older script, spend a few extra minutes to update it according to these best practices
  - technical debt - you can do it quickly or you can do it right. Time saved now may cost you or someone else more time later.

#### 7.10.4.2 Writing functions

http://nicercode.github.io/guides/functions/ Why write functions? * name a chunk of code for easier reading * easily reuse a chunk of code

What makes a good function: * It's short * Performs a single operation * Uses intuitive names

#### 7.10.4.3 Directories and files

```
* Store files in a folder called 'github' in your home directory; access it with `~/github` so that use
```

# 8 Toolbox Troubleshooting

The Toolbox prints messages during its processing to help guide error checking and debugging. Here are a few troubleshooting tips. This section will be updated frequently; please share any problems that you encounter.

## 8.1 General Software Errors

### 8.1.1 rpostback-askpass error

Sometimes when RStudio won't push committed changes to GitHub, RStudio displays an `rpostback-askpass` error:

```
error: unable to read askpass response from 'rpostback-askpass'
fatal: could not read Username for 'https://github.com': Device not configured
```

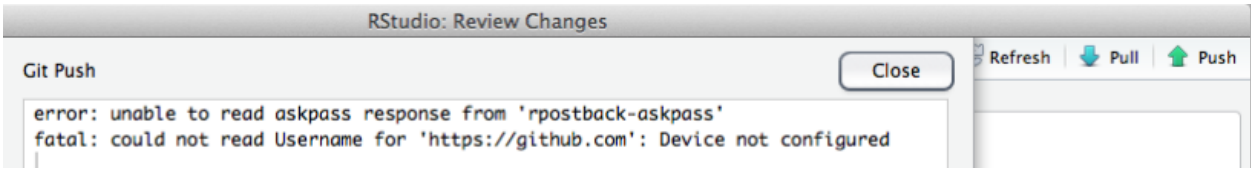
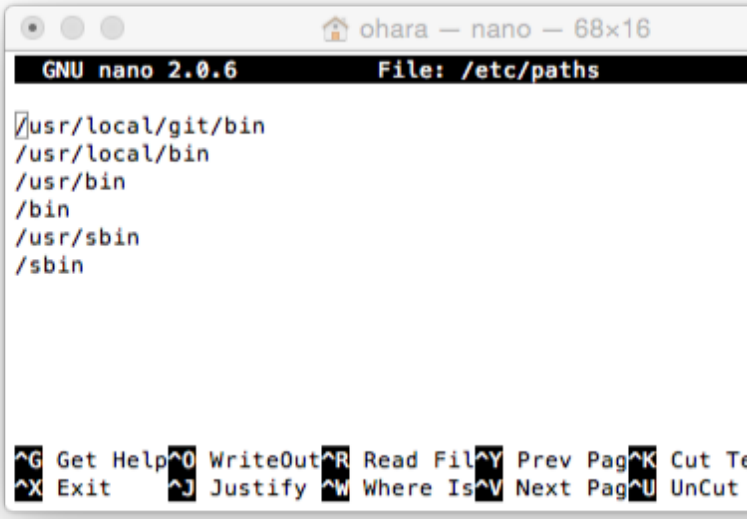



Figure 56: Error screen window: 'error: unable to read askpass response'.

Here's how we fixed it: we updated `git.exe` to the latest version, 2.2.1, edited the search path to point to the new version, made sure the *git* credential.helper was configured to be able to access the OS X keychain, and pushed a test commit from terminal to store the username and password in the keychain, where it can be accessed from other apps like RStudio. Easy peasy!

1. To check your current version of `git.exe`, type this at the terminal command line:
   - `$ git --version` should return something like:
   - `git version 2.2.1` (check online to see if this is the latest version)
2. To update, go to http://git-scm.com/download/mac, download the latest *git* for OS X, install it.
3. In terminal, type `git --version` and verify that it reports the new version. If it shows the new version, great! Skip to Step 5.
   - Don't be sad if it doesn't! If you still see the old version, the installer put the new version into a different directory, which has a lower priority in the search path, so now to update the search path. The default Apple *git* seems to install the `git.exe` into `/usr/bin/` directory, this particular updater seems to install into `/usr/local/git/bin/` directory. The search path needs to be updated to look for `git.exe` in the new directory first.
4. To change the search path, open up the paths file in `nano` editor using `sudo`:
   - `$ sudo nano /etc/paths`
   - At the top line of the paths file, add the directory for the updated *git*: `'/usr/local/git/bin'`(without

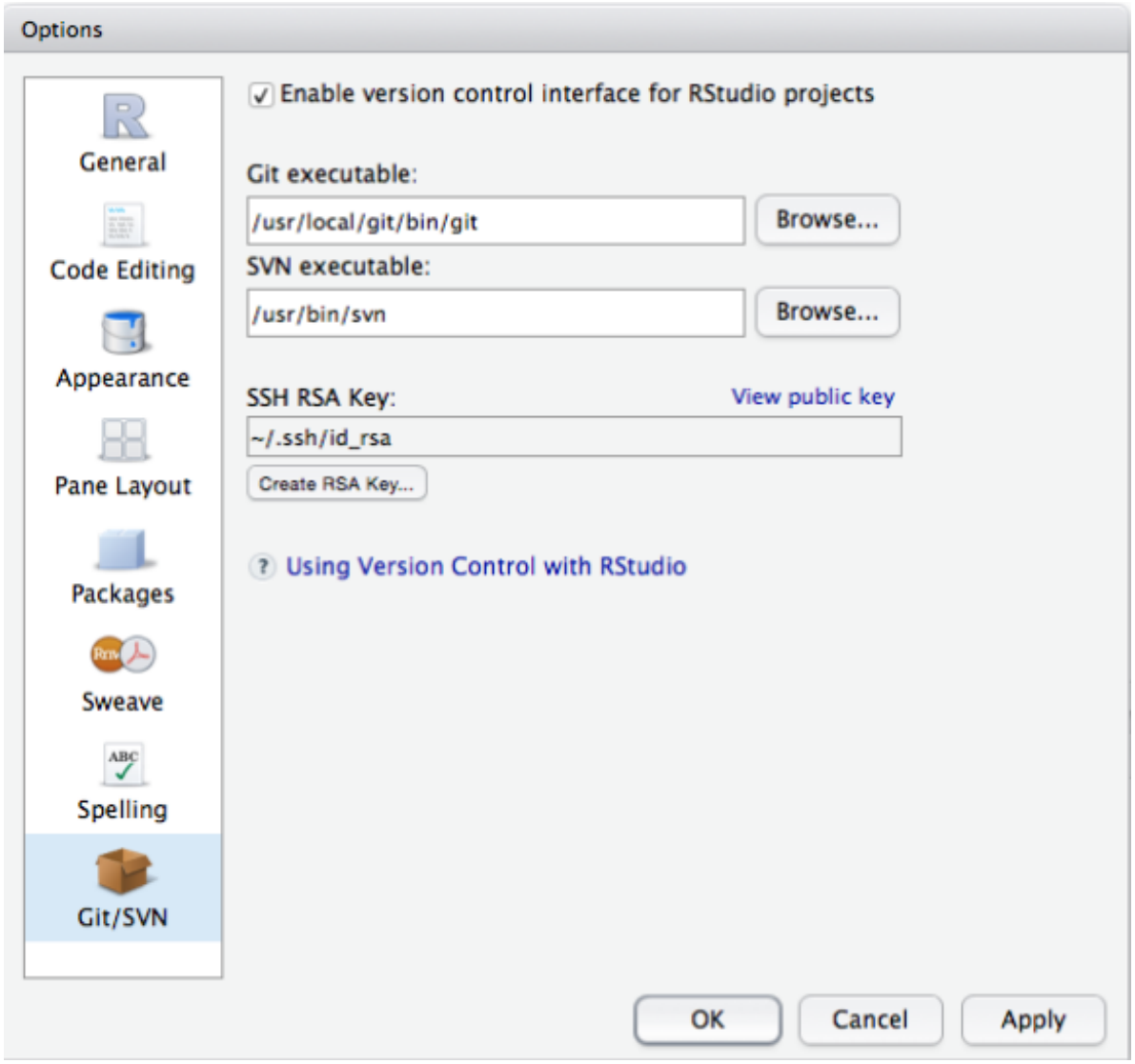



the quotes) so it looks like the top line here:

- Then hit `control-X` to exit, then `Y` in response to the save prompt.

5. Make sure your `git config` is up to date, including `credential.helper`:
   - `$ git config --global -l` should return something like:
   - `user.name="Casey O'Hara"`
   - `user.email=ohara@nceas.ucsb.edu`
     - see https://github.com/OHI-Science/ohiprep/wiki/Setup#git_identity for help on updating user.name and user.email
   - `credential.helper=osxkeychain`
     - (if you need to configure the credential helper: https://help.github.com/articles/caching-your-github-password-in-git/)
6. Now while you are in Terminal, it is important to sync with a repository to establish your security credentials. You must clone a repository and push a 'test' commit, and then once you are prompted for your username and password your information will get stored in the keychain. Here are the steps:
   - Change your working directory to your local github directory: `$ cd github`
   - (Tip: you can check if you're in the right folder by entering `pwd`, short for "print working directory"; or you could look at the line of code preceding the "$".)
   - Clone into a repository with a URL *for which you have permissions*. As an example, the following steps use a repository called 'ZAF' but you should use your own URL with a three-letter country code in place of 'ZAF':
   - `$ git clone https://github.com/omalik/zaf.git`
   - Change your working directory to the folder you just created (here, 'ZAF'): `$ cd zaf`
   - Push a test commit to repository 'ZAF':
   - `$ touch test.md`
   - `$ git add test.md`
   - `$ git commit -m "testing"`
   - `$ git status`
   - `$ git push`
   - Check your status again: `$ git status`
     - (TIP: You can check your status with `$ git status` and you can use 'ls' to see if your new changes have registered in this repository.)

7. Now that *git* is updated and your username and password are set, make sure RStudio knows the location of the new `git.exe`. In RStudio, select **Tools > Global Options...**, select the `Git/SVN`, and browse to the new `Git executable` (it should appear as `/usr/local/git/bin/git` if you updated your *git* version as above).




Next time you push a commit from RStudio, it should remember the username and password from your test commit in Step 6, and you should be good to go.

### 8.1.2 Loading RWorkspace on Restart

When you restart your R Session (**Session > Restart R** on a Mac), if you see that it is trying to load `ohicore`, it may give you an error:

You do not want it to load `ohicore` or to save anything in your workspace. You will need to change the default setting from you **.Rproj** file. Steps to do this:

1. Go to Project Options, either in the pull-down menu or by double-clicking the .Rproj file:

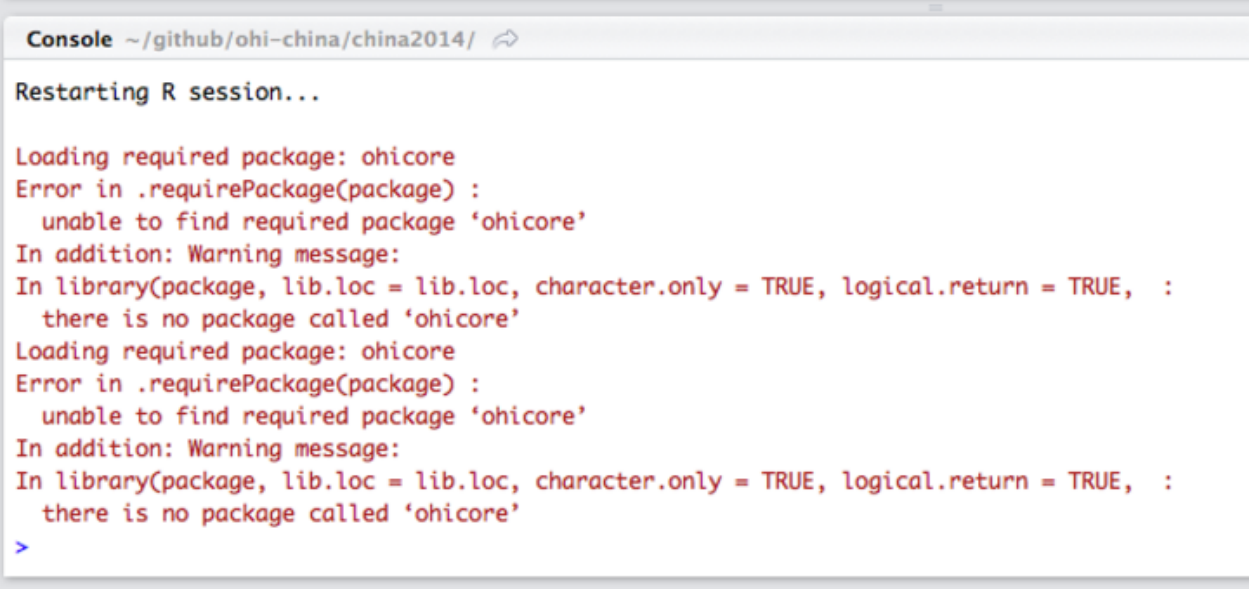



Figure 57:

2. Change all options to **No:**

## 8.2 Errors when Using the Toolbox

### 8.2.1 Useful Errors when Calculating Scores

TIP: You can use the *layers* function in `calculate_scores.R` to error-check whether you have registered your files in `layers.csv` correctly or not. If you haven't, you will get an error message regarding 'missing files'.

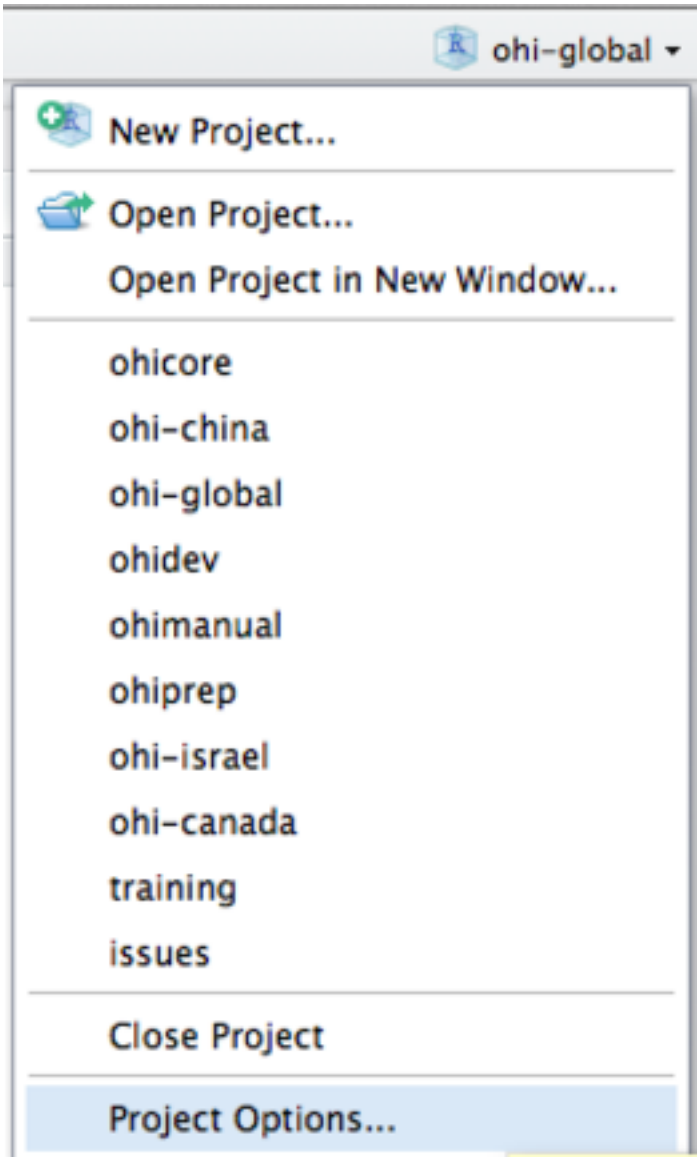

ohi-global
New Project...
Open Project...
Open Project in New Window...
ohicore
ohi-china
ohi-global
ohidev
ohimanual
ohiprep
ohi-israel
ohi-canada
training
issues
Close Project
Project Options...

Figure 58:

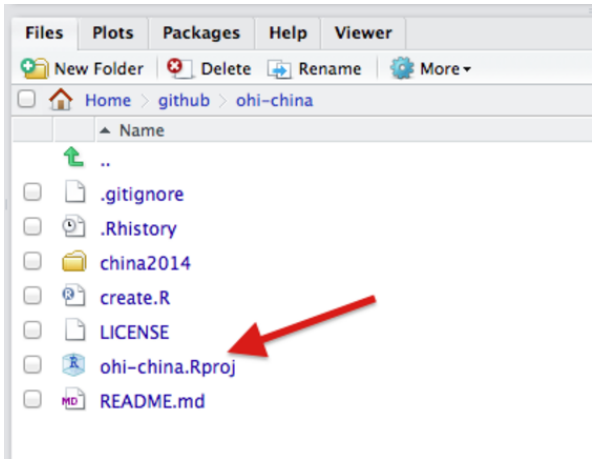
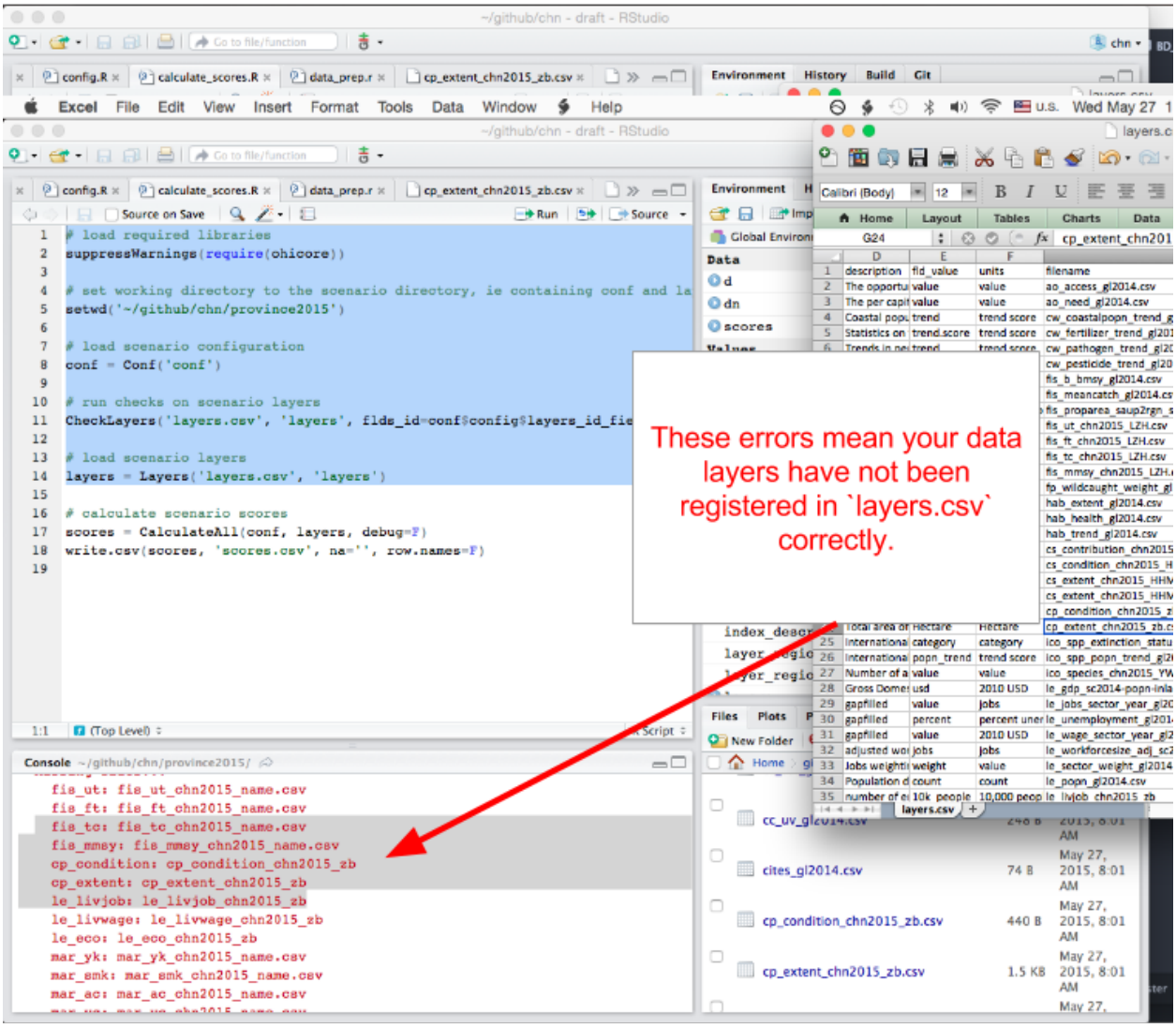

Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
Home
github
ohi-china
Name
..
.gitignore
.Rhistory
china2014
create.R
LICENSE
ohi-china.Rproj
README.md

Figure 59:


These errors mean your data layers have not been registered in `layers.csv` correctly.

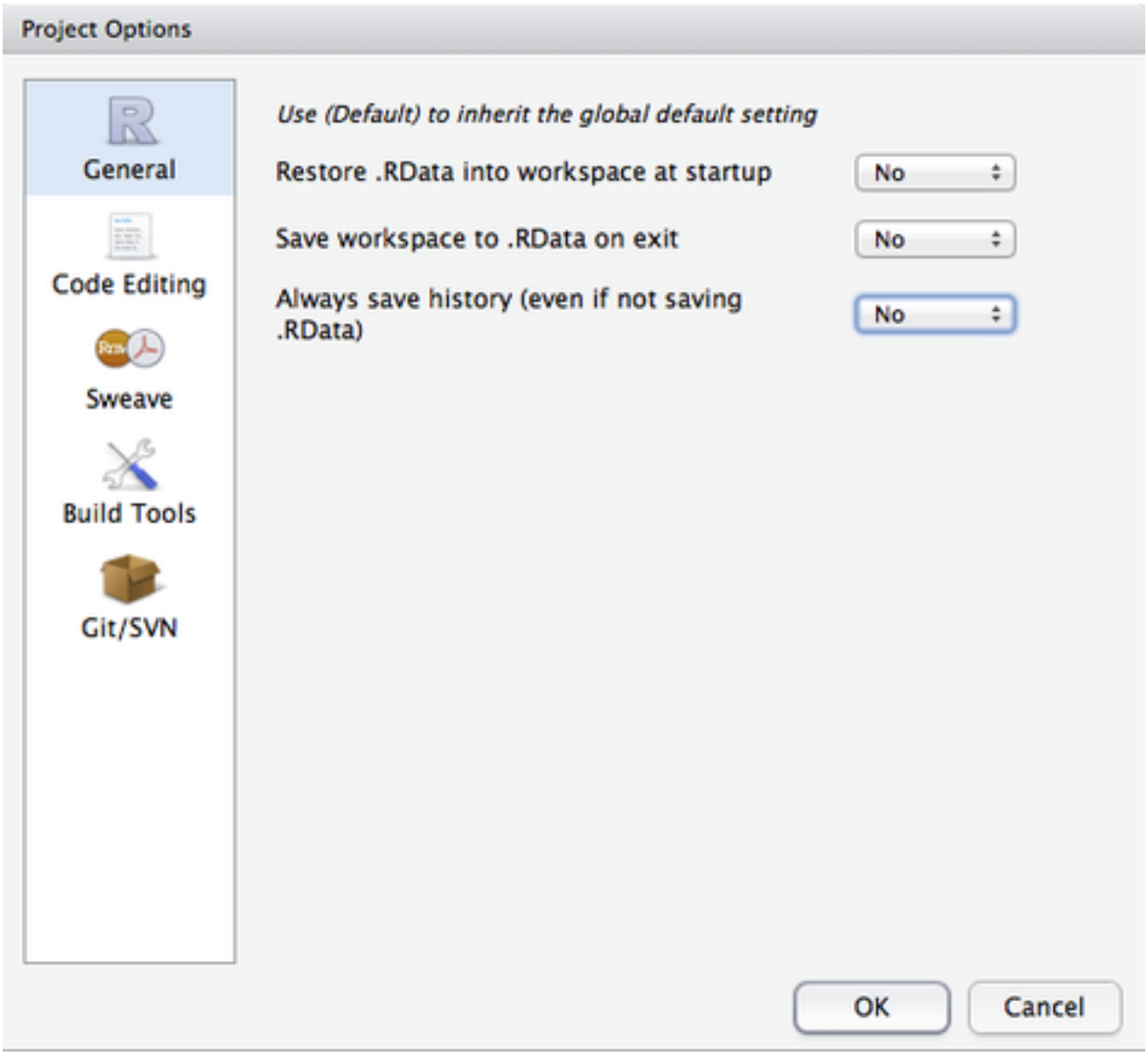

Project Options
General
Code Editing
Sweave
Build Tools
Git/SVN
Use (Default) to inherit the global default setting
Restore .RData into workspace at startup
No
Save workspace to .RData on exit
No
Always save history (even if not saving .RData)
No
OK
Cancel

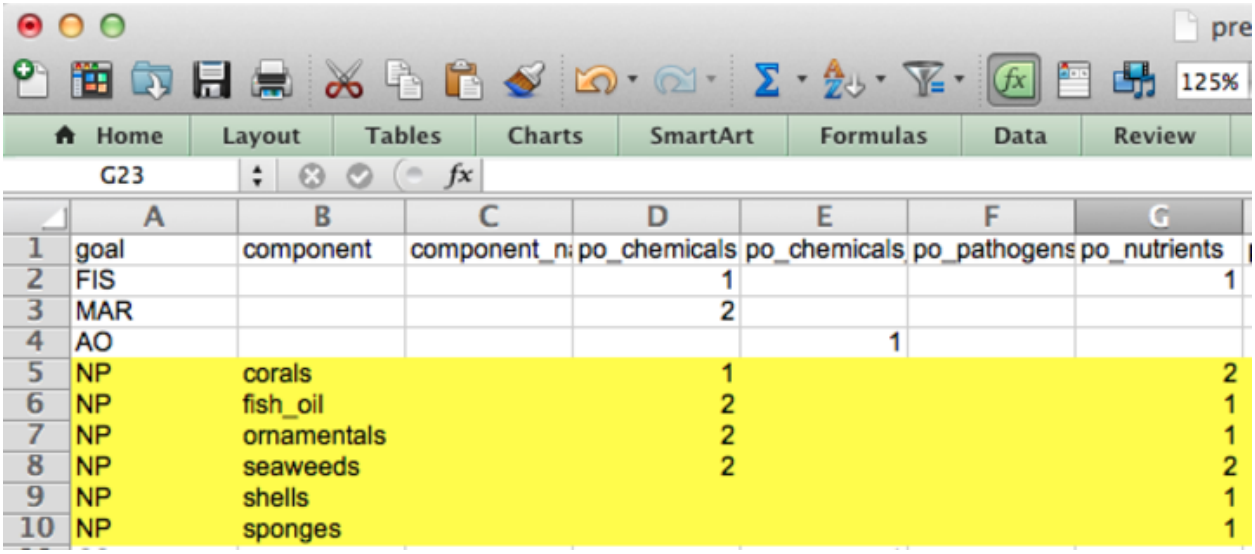
Figure 60:

### 8.2.2 Calculating Pressures...

#### 8.2.2.1 'The following components for [goal] are not in the aggregation layer [layer]...' Example:

```
Running Setup()...
Calculating Pressures...
The following components for NP are not in the aggregation layer np_harvest_product_weight categories (fish_oil,
Error in data.frame(names(P), P) :
  arguments imply differing number of rows: 0, 1
```

Figure 61:

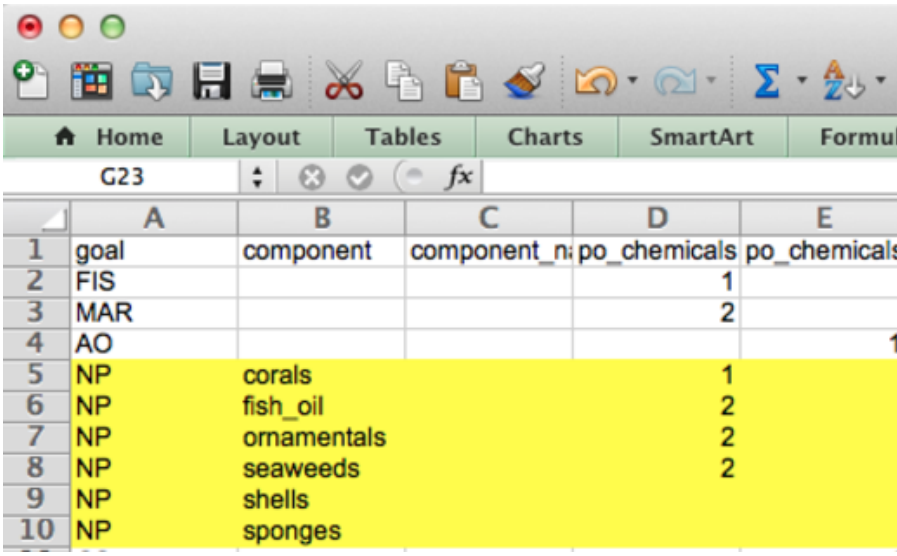



| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | goal | component | component_n | po_chemicals | po_chemicals | po_pathogens | po_nutrients |
| 2 | FIS | | | 1 | | | 1 |
| 3 | MAR | | | 2 | | | |
| 4 | AO | | | | 1 | | |
| 5 | NP | corals | | 1 | | | 2 |
| 6 | NP | fish_oil | | 2 | | | 1 |
| 7 | NP | ornamentals | | 2 | | | 1 |
| 8 | NP | seaweeds | | 2 | | | 2 |
| 9 | NP | shells | | | | | 1 |
| 10 | NP | sponges | | | | | 1 |

Figure 62:

This error means you should update your pressures matrix because it expects there to be components that your region does not have.

#### 8.2.2.2 'Error in matrix. . . ' Example: >




| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | goal | component | component_n | po_chemicals | po_chemicals |
| 2 | FIS | | | 1 | |
| 3 | MAR | | | 2 | |
| 4 | AO | | | | 1 |
| 5 | NP | corals | | 1 | |
| 6 | NP | fish_oil | | 2 | |
| 7 | NP | ornamentals | | 2 | |
| 8 | NP | seaweeds | | 2 | |
| 9 | NP | shells | | | |
| 10 | NP | sponges | | | |

This error means there is an empty column in `pressures_matrix.csv`, and the Toolbox cannot handle empty columns.

### 8.2.3 Calculating Resilience . . .

#### 8.2.3.1 'Error in match(x, table, nomatch = OL) : object id_num not found'

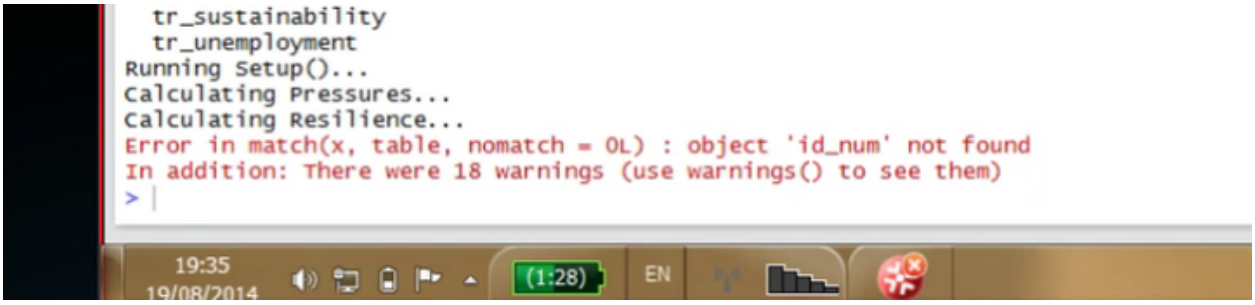



Figure 63:

This error means you should check that there is at least one entry for each goal (for each row) in `resilience_matrix.csv`.

# 9 Frequently Asked Questions (FAQs)

This document provides answers to some frequently asked questions about conducting regional assessments using the Ocean Health Index. A few questions are related to general concepts in the Ocean Health Index, but mostly those topics are covered at http://www.oceanhealthindex.org/About/FAQ/. Here, the FAQ are primarily technical questions regarding regional assessments and using the OHI Toolbox. This document will be updated continually as we have more questions. Questions are arranged by theme, and have the format Q: (question) and A: (answer).

## 9.1 Overall

## 9.2 Conceptual

**Q: Are regional assessment scores comparable with global assessment scores?**

A: Regional Index scores cannot be directly compared to global Index scores, or to other regional Index scores calculated through separate efforts. This is because data and indicators (both what they measure and their quality), reference points (set using local knowledge and priorities), and specific goal models are often different for the areas being compared.

However, because scores for each goal are scaled to a reference point, qualitative comparisons can be made. For example, a score of 71 in the US West Coast compared to 66 in Brazil says that the US West coast is closer to fully meeting its sustainable goals (i.e., meeting regional reference points). Furthermore, use of the same Ocean Health Index framework across regional assessments permits fruitful discussion and general comparisons even if data inputs differ. Ocean Health Index assessments at any scale always work within a standardized definition of ocean health, using information to capture the philosophy of the ten goals that have been identified (and undergone scientific peer-review) prior to compiling relevant data. Use of the ten-goal framework is important both to ensure that all aspects of ocean health are captured and to allow better comparison across regional assessments than would be possible if the different regions used different methods.

**Q: How does the Index account for ecosystem benefits?**

A: The OHI is not an index of ecosystem services. The Index prefers to describe benefits from a healthy ocean and emphasize their relevance, but the ideas are closely related. The ten goals roughly fall into areas of ecosystem services such as food provisioning (**Food Provision**), regulatory services (**Carbon Storage**), cultural services (**Tourism and Recreation**,**Special Places**), supporting services (**Clean Waters**, **Biodiversity**), and other values (**Livelhoods and Economies**).

*(Source: OHI Baltic workshop)*

**Q: Where is climate change measured in the Index?**

A: Four different aspects of climate change – increases in sea surface temperature (SST), sea level rise (SLR), ultraviolet radiation (UV), and ocean acidification (OA) – are included as pressures to many goals in the Index, including Natural Products, Carbon Storage, Coastal Protection, Sense of Place, Livelihoods & Economies and Biodiversity. Mitigation of climate change through carbon storage is one of the ten goals.

**Q: Why are food provision and artisanal fishing opportunities goals separated?**

A: These goals measure different aspects of how people relate to fishing. The catch of fish made by artisanal (=small-scale, subsistence type) fisheries is captured in the food provision goal. Jobs, wages and income from both the food provision and artisanal fishing goals are captured in the livelihoods & economies goal. The purpose of the artisanal fishing opportunity goal is to evaluate the opportunity for people to pursue this fishing in relation to their need to do so.

## 9.3 Timing and Resources

**Q: How much does it cost to produce a regional assessment?**

A: Regional assessments can be completed at(varying costs depending on the local context.(Funds are needed for a management and scientific team, workshops and meetings (including travel), communications, policy engagement, and operating costs. Therefore, securing funding is an important component to satisfactorily complete the assessment. We encourage the development of a local proposal or strategic action plan that details a timeline of activities and the resources needed to accomplish them.

**Q: How many people are required in a team?**

A: rather than a specific number of individuals, what is required are specific skillsets. For example, if the scientific analysts were capable of effectively conducting the R analysis, then a dedicated R analyst would not be required. In current assessments, teams range between 2 and 8 people.

**Q: How long does it take to calculate OHI at a regional scale?**

A: The duration of an OHI assessment depends on a number of factors, such as the budget and number of people involved, the scale of the study area and whether new regions will need to be created, how easily data can be acquired, how much local data can be incorporated, how many goal models need to be changed. Additionally, decisions about setting reference points require input from experts. For independent assessments (OHI+), we have found that the average time has ranged from 1.5 to 3 years (See **Task Timeline** in the **Conceptual Guide**).

**Q: How much time will modifications by an R analyst take?**

A: This will depend on if you are changing any models, and potentially data layers–but a lot of changing data layers just requires registering them properly in `layers.csv` (and maybe `pressures_matrix.csv` and resilience_matrix.csv if they are pressures or resilience files) and having the `functions.R` file call those layers. That is more 'bookkeeping' than actual R programming.

**Q: How much time will modifications by a GIS analyst take?**

A: this will depend on how many layers you are processing: you are clipping spatial data? That will take some time because there are quite a few files, but maybe not too long since it is pretty small scale and once there is a clipping mask created I think you apply it to other files.

**Q: Which goals require a GIS analyst?**

A: All goals using spatial data could potentially require a GIS analyst. These goals are commonly: habitat-based goals and sub-goals: (Coastal Protection, Carbon Storage, Habitats—a sub-goal of Biodiversity), Food Provision, Sense of Place, Species—a sub-goal of Biodiversity, Clean Waters

## 9.4 Structure

**Q: Can we remove or add goals to the OHI?**

A: A lot of deliberation went into defining the ten goals, and they seem to do a pretty good job of covering many if not most ocean uses, so additional goals may not be necessary. But it could be that they eclipse or replace an existing goal.

## 9.5 Reference points

**Q: Can planning targets can be used as the reference points?**

A: Yes, planning targets can be used as reference points. This won't be appropriate for every goal, but there are cases where this seemed best (example: iconic species sub-goal in the global assessment, mariculture sub-goal in the US West Coast assessment).

**Q: What is sector evenness?**

A: Sector evenness (also called a diversity index) is an economic concept that is included in OHI to enable comparison across many different sectors included in the Livelihoods & Economies goal. This goal evaluates jobs, wages and revenues for nine marine employment sectors. The distribution of employment across these nine sectors is an effective indicator of resilience. If total employment within a community is primarily based in one or two sectors, the overall economic system will be excessively vulnerable to downturns in those sectors. Conversely, if employment is spread relatively evenly throughout all nine sectors, the overall system will be more robust and resistant to such disturbances. Overall revenue within the community will remain more stable during such downturns, and workers displaced by a downturn in their sector may be able to find employment in another sector without leaving the community.

## 9.6 Appropriate data layers

**Q: Shipping and port activity are hardly affected by the health of the ecosystem. Why are these included in the Index?**

A: Shipping and port activity are included as pressures only

**Q: Can oil spills be included in OHI?**

A: Yes, oil spills could be included as a pressure and in the Clean Waters goal.

**Q: Is seasonal (non-permanent) sea ice included in OHI habitats?**

A: No, sea ice only includes permanent sea ice.

**Q: Can seaweeds be included in the Carbon Storage goal?**

A: Because they store carbon for less than 100 years, seaweeds and corals are not included in the carbon storage goal. While the pelagic oceanic carbon sink (phytoplankton) plays a large role in the sequestration of anthropogenic carbon, the pelagic ocean mechanisms are not amenable to local or regional management intervention. Phytoplankton and contribute to carbon fixation when they die and sink to the sea bottom at sufficient depth, because it is effectively out of circulation. However, if those phytoplankton are eaten, the carbon is cycled back into the system and not sequestered. Something that could potentially be included in the carbon storage goal is mollusc shells, if they are added to a landfill and not recycled in the sea. So if information on mariculture production and waste disposal are available, this could be an interesting addition to carbon storage at a regional scale.

**Q: Is coastal engineering included in Coastal Protection? What if it reduces erosion?**

A: We did not include an assessment of the protection afforded by man-made structures, such as jetties and seawalls, because these structures cannot be preserved without maintenance, may have other negative side effects (e.g. alter sedimentation rates causing erosion in new locations), thus they do not constitute long-term sustainable services. Coastal engineering (jetties, harbours, marina and breakwater) is not natural, and is mostly seen as a pressure. It will also be evident in the status of due to decreased natural habitat. It gets tricky when structures are built to help reduce coastal erosion–they are still manmade and therefore not a natural benefit that the ocean provides. But if available data allow, it might be possible to include tradeoff effects: maybe in areas where natural habitats are degraded and man-made structures have been built to reduce erosion, we could reduce the pressure that would otherwise be applied.

**Q: How is seawater used for cooling on-shore power plants incorporated into OHI?**

A: The use of cooling water for on-shore power plants would be a pressure on the ocean, since it causes entrapment of fishes, larvae, etc, and usually is circulated back into the ocean at higher temperatures (and maybe other chemicals, minerals, etc). Since the energy is coming from land-based activities, there isn't a service that the ocean is providing that 'benefits' people, it is only a pressure from the OHI perspective.

**Q: How is freshwater production through desalination incorporated into OHI?**

A: Desal would be incorporated into OHI in several places. The benefit is that there is freshwater produced, which could be incorporated into the Natural Products goal (or potentially into its own goal). Data required would be the volume of freshwater created based on the volume of seawater involved and spatial extent. Setting the reference point would not be based on how much can be produced, but some other targets perhaps set by government (percentage of the population served). Similar to the mariculture sub-goal and tourism goals, any negative effects caused by desal that affect other goals (example: species) do not influence the ability to obtain desalination targets now and in the future. Therefore, the sustainability coefficient only measures the ability to sustain that goal, but not the impacts on other goals: instead, they are taken into account as pressures when calculating the other goals. Desal should be included as a pressure similar to cooling on-shore power plants since the discharge brine is dense, doesn't plume very well and there are chemicals involved.

**Q: Where do energy activities fit in to OHI?**

It depends. Energy could be part of a **Natural Products** goal, for instance, such as wave energy – but then the question is, what is the reference point? It is partially accounted for in **Livelihoods & Economies** through sectoral jobs data. The infrastructure is also something to consider. It could also be a pressure or resilience factor if there is a measurable footprint of the activity. You may want to consider for resilience, do you have governance measures that promote more sustainable practices in the energy industry?

## 9.7 Food Provision

**Q: Could the culture of marine fish in closed pools on-shore be included in the Mariculture sub-goal?**

A: This should not be included because onshore aquaculture does not require a marine environment.

**Q: Can aquaculture farms that receive seawater supply and return seawater back to the sea be included in the food provision goal?**

A: This would be more appropriately included in the Mariculture sub-goal, and with finer-scale data additional pressures due to the intake pipes and the processed brine back into the marine system could be incorporated as well. Natural Products

**Q: If natural products are all produced through on-land aquaculture, should this goal be removed?**

A: In this case you would probably have good reason to exclude the natural product goal due if this was defendable through discussions with experts and any reports/papers on the topic. This would also depend on the origin of these natural products–are they from the region's waters? Habitat-based goals

**Q: I have fish that are used as feed for other fish (e.g., sprat) in my country. Can I include them in this goal?

A: It would be more appropriate to include them in Natural Products rather than Food Provision. This is because they are not being consumed directly. Fish such as sprat, for example, may be used to feed pigs in addition to other fish, and therefore you would need to know how much (tonnage) is being produced, and where it is going to be able to accurately distinguish these categories to avoid double-counting.

*(Source: OHI Baltic Workshop, February 2015)*

**Q: How is coral health calculated?**

A: Coral health was estimated by compiling point data from multiple studies of percent live coral cover. In other words, estimates of coral cover within transects of certain sites were repeated in time and we used that rate of change in time as an indication of health of the reefs in the whole region. The difficulty lies in 1) having enough different locations sampled that you can say something about the whole region and 2) finding studies that did repeated measures in time, in the same location, over at least 20 years. In the Global 2013 assessment, there were so few datasets that satisfied this condition that we had to pool observations from different locations.

**Q: Is it possible to calculate habitat goals when there is only one year of habitat data?**

A: With only one year of habitat data, it is not possible to calculate the trend (which requires 5 years of data). Instead, it might be best to use the available habitat data to calculate the current status and then to overlay pressures for the last 5 years to calculate trend.

## 9.8 Livelihoods & Economies

**Q: Benefits gained from Wild-caught fisheries, Mariculture, Tourism & Recreation are included in specific goals. Why are these counted again in Livelihoods & Economies?**

A: The quantity of fish, mariculature, and participation in T&R are considered separately in goals whereas the monetary component is captured in L&E.

**Q: Why are revenue data from shipping, boat building, ports and harbors included as revenue? Do these activities rely on a healthy ocean?**

A: These sectors are included in the Ocean Health Index because the demand for some of those boats (fishing boats, sailboats, yachts) is dependent on a healthy ocean.

**Q: Why isn't oil and gas industries included in revenue?**

A: The Natural Products goal does not include non-living items such as oil, gas, and mining products, because these practices are not considered to be sustainable. They are also done at such large scales that including them would essentially make OHI an index for oil and mining–and they are not truly an ocean product. Because these products are not included in terms of quantity extracted, it did not seem appropriate to include information regarding jobs, wages or revenue.

## 9.9 Tourism & Recreation

**Q: How do I calculate the sustainability term for TR?**

A: The best way is to use a local indicator or measure of tourism sustainability or competitiveness, otherwise use the TTCI value from the Global 2013 assessment for the study area (applied evenly across all regions.

## 9.10 Natural Products

**Q: Where do Natural Products come from?**

A: In the global assessments, Natural Products data come from the UN's Food and Agriculture Administration (www.fao.org/fishery/statistics/software/fishstatj/en). These data are compiled and reported by product for each country, and available by downloading the FishStatJ software.

## 9.11 Species

**Q: Can species and iconic species model scores be penalized if there are local flagship species that have not been evaluated?**

Global data are based on IUCN assessments. For these evaluations, IUCN chooses a taxon (e.g. sharks) and a group of world experts assess it comprehensively. Locally identified species identified in a regional assessment may not be in the IUCN database because they do not belong to one of the taxa that have been selected for assessment, or because the experts that did the assessment did not know that information existed. In either case, there is no connection between what IUCN reports and what assessments are done locally. Therefore, it might not be fair to penalize a study area for missing species. For biodiversity, it is unrealistic to expect that all species are assessed, so it seems unfair to penalize for unassessed species. In the fisheries goal, there are penalties for species that are exploited but not assessed, because if there are landings data, it means they are somewhat measurable, and so it is reasonable to expect they should be at least monitored.

It might be reasonable to penalize unassessed iconic species. It is a smaller list of species that are specifically identified as being of interest, for one reason or other. This would work for species that have some form of assessment - unless that information already exists, it might be unrealistic to try to produce the data layer required to develop a new model.

## 9.12 Sense of Place

**Q: Data are only available for marine protected areas, not terrestrial protected areas. Can we still calculate the Lasting Special Places sub-goal?**

A: Yes, it is possible to calculate only the marine component of this sub-goal: this is not ideal but OHI is flexible to work with the data available.

**Q: Should we calculate each category used in our assessment (e.g., antiquities, MPAs, beaches of special interest) independently, and then give the same weight (e.g., a third of the goal score) to the three categories, or should we instead pool the actual areas of the 3 categories?**

A: Whether you group them together or calculate each category separately depends on reference points. Maybe you want 10% of offshore water to be in MPAs, but only 5% of coastlines to be beaches and 3% Antiquities, for example; in this case, you would calculate them separately and then add them together. But if you want 10% of your country's coast to have any combination of these things, you would keep them together.

*(Source: OHI Israel assessment discussions, 2014-2015)*

## 9.13 Pressures

**Q: How are single ecological pressures (si in Equation S8) calculated?**

A: Data included in pressures calculations are accessed in the same manner as any other data layer, and rescaled from 0-1 with an appropriate reference point. For further information, see HowTo_GatherAppropriateData and HowTo_CalculatePressures from ohi-science.org.

**Q: Does the pressures matrix need to be changed?**

A: It is likely that the pressures matrix will not need to be changed. The weights assigned in the matrix were set using information from the literature and by experts; the matrix was created by Halpern et al. 2012.

**Q: How is commercial high and low bycatch calculated?**

A: Commercial high and low bycatch are categorical values that were set based on fishing gear type. This began as a list of gear types used, producing a range of potential bycatch frequencies (from local reports when possible), which can be rescaled.