# Template for pandoc / markdown manuscripts

Timothée Poisot     Second Author

February 5, 2018

**Abstract**: This template uses the magic of makefiles, pandoc, and markdown, to make it easy to produce multiple documents from markdown, R markdown, or Julia markdown files. Just type `make` at the command line to see the different options.

**Keywords**: ecological networks - beta-diversity - biogeography

Timothée Poisot *1,2* Second Author *2,3*

**1**: Université de Montréal, Département de Sciences Biologiques; **2**: Québec Centre for Biodiversity Sciences; **3**: University of Whatever

Correspondence to Timothée Poisot – `timothee.poisot@umontreal.ca`

This project intends to make the generation of high-quality preprints from markdown, R markdown, and Julia markdown documents easy. Once downloaded, type `make` to see the output. This will generate two pdf documents and one OpenDocument file.

## 1 Installation

To get started, you will need the python `pandoc-fignos`, `pandoc-eqnos`, and `pandoc-tablenos` filters.

```
make dependencies
```

Make sure that `pandoc` and `pandoc-citeproc` are installed, and that you have a LaTeX installation. You will also need an installation of node. If you want to use this template with reproducible documents, you will need either `knitr` or `Weave.jl`.

## 2 Document options

There are two important files to edit to specificy the manuscript informations. First, `authors.yaml` should be self-explanatory; it contains the author names, email addres for the corresponding author, and affiliations. The `infos.yaml` file is for the manuscript title, keywords, etc. Finally, the `ABSTRACT` file has the abstract. It can contain markdown formatting.

**2.1. Tables** Table legends go on the line after the table itself. To generate a reference to the table, use `{#tbl:id}` – then, in the text, you can use `{@tbl:id}` to refer to the table. For example, the table below is table 1. You can remove the *table* in front by using `!@tbl:id`, or force it to be capitalized with `\*tbl:id`.

**Table 1** This is a table, and its identifier is `id` – we can refer to it using `{@tbl:id}`. Note that even if the table legend is written below the table itself, it will appear on top in the compiled document.

| Using | produces |
|---|---:|
| `@tbl:id` | table 1 |
| `!@tbl:id` | 1 |
| `\*@tbl:id` | Table 1 |

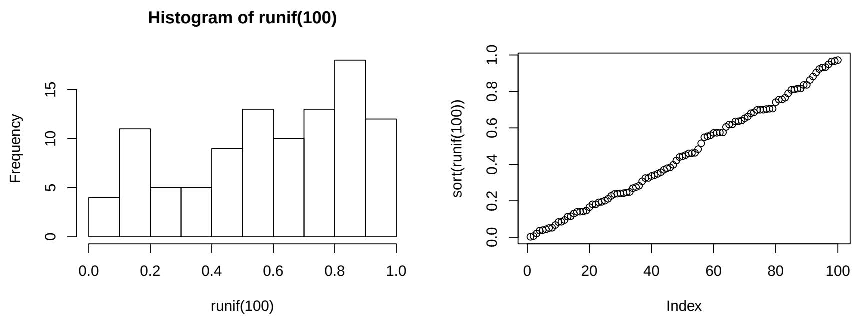

**Figure 1** This is a figure. Figures can have identifiers, and the width can be changed as well. This legend is a bit long, to show what happens in the preprint mode (it continues in the margin below the limit of the figure).

**2.2. Equations** Equations can be referenced using the same syntax as tables, using the `eq` prefix in place of `tbl`. For example:

$$y = mx + b \tag{1}$$

We can refer to eq. 1 in the text.

**2.3. Adding references** References go in the `references.json` file, at the root of the project. References are cited with `@key`, where `key` is the unique identifier of the reference. Both inline, like Hutchinson (1959), and in brackets (Hutchinson 1957) can be used.

You can also have footnotes.[1]

(1) this is a footnote – it is actually rendered as a sidenote in the preprint format.

**2.4. Figures** Figures can be used with the usual markdown syntax. After the path, you can use `{#fig:id width=50%}` to specify the width and the reference. See table 1 for how to cite. The code below in the markdown source produces fig. 1.

## 3 Other elements

**3.1. Code blocks** You can use fenced code blocks to render code:

```
// Update affiliations
var print_affiliations = []
for (var af in affiliations) {
  var afobject = {}
  afobject.id = affiliations[af]
  afobject.text = af
  print_affiliations.push(afobject)
}
```

Note that code blocks have line numbers of the left, so this does not interfer with the line numbers of the text (which are on the right).

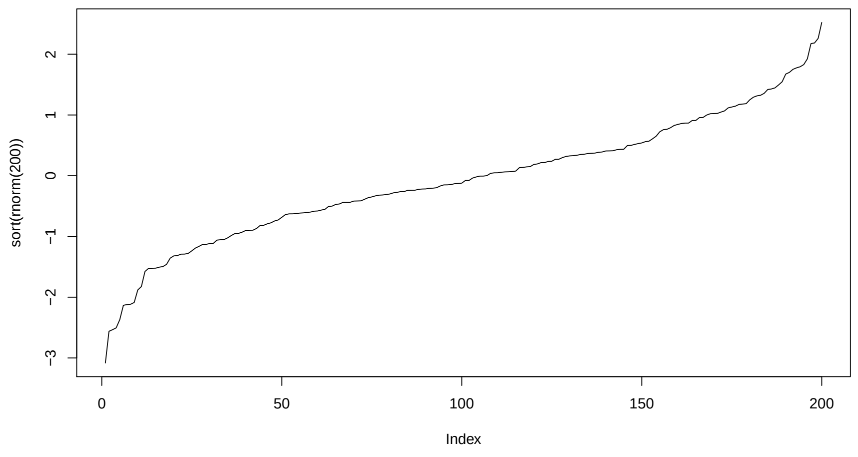

**Figure 2** This is the figure created by the chunck `testfig`, so it is in `figure/testfig-1`. You can use different `dev` in the knitr chunk options, so it is possible to generate pdf or png figures.

### 3.2. Track changes

You can use `make diff` to create a marked-up pdf document. The git revision can be specified with the `TAG` variable of `make` (by default, the latest commit). The other option is `AS`, which can be `draft` or `preprint`, to render the marked-up version as a draft or as a preprint.

### 3.3. Editorial marks

Critic Markup is rendered:

Don't go around saying~~to people that~~ the world owes you a living. The world owes you nothing. It was here first. ~~One~~Only one thing is impossible for God: To find any sense in any copyright law on the planet. Truth is stranger than fiction[strange but true], but it is because Fiction is obliged to stick to possibilities; Truth isn't.

Note that CriticMarkup is *not* rendered into OpenDocument.

### 3.4. Using with knitr, Weave.jl, …

Just type `make`. If there is a `Rmd` or `Jmd` document with the same base name, the makefile will render the markdown document for you.

Note that the extensions *must* be `Rmd` or `Jmd`, with an uppercase first letter. Of course you will need `knitr` (for `R`) or `Weave.jl` (for `julia`).

Because of the way figures are refered to (using the `@fig:id` syntax), it is better to generate the figure first, and then call it in the text, using `fig.show='hide'`. The code below will generate fig. 2.

```
plot(sort(rnorm(200)), type='l')
```

You can then use this figure:

With `knitr`, the `kable` function can create tables. If you add the caption paragraph immediately below, then these tables can be cited. This is how we produce table 2.

**Table 2** This is a table, and its identifier is `knit` – we can refer to it using `{@tbl:knit}`. Note that even if the table legend is written below the table itself, it will appear on top in the compiled document.

| Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|---:|---:|---:|---:|:---|
| 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 5.0 | 3.6 | 1.4 | 0.2 | setosa |
| 5.4 | 3.9 | 1.7 | 0.4 | setosa |

## 4 Text example

We posit that four simple rules govern the evolution of networks. First, every network originally consists of just two species sharing a single interaction; for example, a plant and its herbivore. Second, a speciation event happens at the top level (*e.g.* the herbivore) with probability $p$, or at the bottom level with probability $1 - p$. Third, the incipient species starts with all interactions of its ancestor. Fourth, some of these interactions are lost with probability $\varepsilon(\lambda, k, c)$, which allows interactions—that are gained through speciation—to be lost either at a fixed rate $\lambda$ or as a function of the incipient species' degree $k$. The $c$ parameter modulates this relationship further by influencing whether high degree of an ancestor increases, or decreases, the probability of the incipient species losing interactions. We have used the following formulation for $\varepsilon$:

$$\varepsilon(\lambda, k, c) = \left(1 + \left(\frac{1}{\lambda} - 1\right) \times c^{k-1}\right)^{-1} . \tag{2}$$

In this formulation, $k$ is the number of interactions of the incipient species, $\lambda$ is the *basal* rate of interaction loss, and $c$ is a parameter regulating whether species with more interactions tend to gain or lose interactions over time. Negative values of $c$ imply that *rich get richer*, *i.e.* species with more interactions tend to conserve them more over speciation. The special case of $c = 0$ corresponds to no relationship between the degree of a species and its probability of losing or retaining an interaction over speciation. The resulting probability of interaction loss, and its consequences on degree, is shown in figure. The values of $\varepsilon$ belong to $]0; 1[$. Note that, because species are duplicated upon a speciation event, the network still grows over time. If an incipient species should lose all of its interactions, then it fails to establish.

These four rules translate directly into steps for the model: pick a level at random, select a species to duplicate, assess the survival of interactions of the incipient, and add the incipient to the network. These are performed a fixed number of time – we impose an upper limit to the richness at each level, and when this limit is reached, the incipient species replaces one of the resident species at random. An equilibrium for the measures of network structure (see next section) is reached within 1000 timesteps. For all situations, we recorded the network after 5000 iterations.

### 4.1. Network measures

*(4.1.1) Connectance* Connectance, defined as the ratio of realized interactions on the total number of potential interactions, is one of the most common descriptor of network structure. In a bipartite network with $T$ species at the top, and $B$ at the bottom, having a total of $L$ interactions, it is defined as $Co = L/(T \times B)$. Connectance has a lower bound, as the network cannot have fewer interactions that the number of species in its more speciose level – the minimal connectance is therefore $c_m = \max(T, B)$. This makes the connectance of networks of different sizes difficult to compare, especially since bipartite networks tends to have a low connectance. For this reason, we used a corrected version of connectance, defined as

$$Co^{\star} = \frac{L - c_m}{T \times B - c_m} \,. \tag{3}$$

This takes values between 0 (the network has the minimal number of interactions) and 1 (all species are connected), but is robust to variations in species richness.

## References

**Hutchinson**. (1957). Concluding remarks. *Cold Spring Harb Symp Quant Biol.* 22:415–27.

**Hutchinson**. (1959). Homage to Santa Rosalia or why are there so many kinds of animals? *Am Nat.* 93:145.