Dashboard Documentation

Circular Risk Model

Model version: 0.1

31/07/2023

Financial Risk Management

Table of Contents

[Table of Contents 1](#_Toc141703211)

[1 Introduction 1](#_Toc141703212)

[1.1 Model purpose 1](#_Toc141703213)

[1.2 Specification and Scope 1](#_Toc141703214)

[2 Dashboard Creation 2](#_Toc141703215)

[2.1 Chosen Frameworks 2](#_Toc141703216)

[2.2 Set up the Dashboard 2](#_Toc141703217)

[2.3 Connect Google Sheet 3](#_Toc141703218)

[3 Risk Score Calculation 3](#_Toc141703219)

[4 Templates 4](#_Toc141703220)

[4.1 Conclusion 4](#_Toc141703221)

[Appendix 5](#_Toc141703222)

[Appendix A: Link to Excel file 5](#_Toc141703223)

[Appendix B: Documents Received 5](#_Toc141703224)

[Appendix C: Reference List 5](#_Toc141703225)

# Introduction

This document describes the process of developing the dashboard for modelling the risk of circular economy companies. The dashboard is available [here](https://dasbhoard-7sgf0oydm6l.streamlit.app/)[[1]](#footnote-1). The code is available on [this Github](https://github.com/irisreitsmaRQ/dasbhoard)[[2]](#footnote-2).

## Model purpose

The dashboard for the circular risk model outputs a risk score given input parameters corresponding to a circular economy company. This score can be used as a decision factor in deciding which circular economy deals are accepted or rejected by, for example, banks. In addition, the dashboard can give insight into which risk factors are important in making these kinds of decisions.

## Specification and Scope

The scope of the dashboard is to test a first draft of the circular risk model (v0.1). This draft can be used to obtain preliminary results and to test the model. The obtained feedback and results can be used to create a second draft of the model (v0.5), which can be tested by a selection of companies. After evaluation of the second draft, the final version of the circular risk model and dashboard can be created (v1.0).

# Dashboard Creation

This chapter describes how the dashboard was created and hosted, and how user inputs are stored.

## Chosen Frameworks

**Streamlit:** The dashboard is created using [Streamlit](https://streamlit.io/)[[3]](#footnote-3), which is an open-source app framework that makes it very easy to quickly create good-looking dashboards using Python. Additional advantages of Streamlit are that their platform makes it possible to host dashboards for free, and that it works very well with [Github](https://github.com/)[[4]](#footnote-4).

**Google Sheets:** The user inputs from the Streamlit dashboard are stored in a [Google Spreadsheet](https://www.google.com/sheets/about/)[[5]](#footnote-5). The first reason for this decision is that Google Spreadsheets works well together with Streamlit. The second reason is that Google Spreadsheets are easily shareable or exportable to CSV or Excel format.

## Set up the Dashboard

Creating a Streamlit dashboard and hosting it on their platform only needs a couple of steps:

1. **Create an account for [Streamlit Cloud](https://streamlit.io/cloud)[[6]](#footnote-6).
2. Connect the account to a Github account by clicking
3. Create a Github repository for the Streamlit app.
4. Create a Python script in the repository, make sure [Streamlit is installed](https://docs.streamlit.io/library/get-started/installation)[[7]](#footnote-7). A simple script could look like this:

# *simple\_script.py*

import streamlit as st

st.write(‘Hello world!’)

user\_input = st.text\_input(label=’Say something’, value=’Something’)

This script can be run locally by executing the following command:

streamlit run simple\_script.py

**If it works, a tab displaying the dashboard opens in the web browser.

1. Deploy the app on Streamlit Cloud by clicking

If it works, a tab will open in the web browser displaying the dashboard.

For more information and a more elaborate explanation, please refer to [this blog](https://blog.streamlit.io/host-your-streamlit-app-for-free/)[[8]](#footnote-8) about hosting a Streamlit app or [this documentation](https://docs.streamlit.io/library/get-started/create-an-app)[[9]](#footnote-9) for creating a Streamlit app.

Now that the dashboard is working, it can be connected to a Google Spreadsheet to store user input from the dashboard.

## Connect Google Sheet

Connecting the Streamlit dashboard to a Google Sheet takes some more steps that are less straightforward. Please follow the steps in [this Streamlit documentation](https://docs.streamlit.io/knowledge-base/tutorials/databases/private-gsheet)[[10]](#footnote-10). These steps are summarized in the subsections below.

**IMPORTANT:** The Streamlit documentation uses Python package gsheetsdb. However, this package is outdated, so please use gspread instead, see also Section 2.3.4.

### Set up Sheet and API

First of all, a sheet needs to be created and the API needs to be enabled in order to read from and write to the sheet from Python:

1. Create a Google Sheet.
2. Enable Sheet API.
3. Create a service account.
4. Download key file from the service account.
5. Share the Google Sheet with the service account. Please choose the correct permission: for just reading the data in Python, the *Viewer* option is sufficient. For writing input from the dashboard to the sheet, the permission should be *Editor*.

### Set up Secrets file

The local Streamlit application reads the sheet URL and other information from a .streamlit/secrets.toml file. This folder and file should be created in the repository if they are not already there. The file should contain the URL to the sheet and the information from they key file downloaded (below the box [gcp\_service\_account]):

# *.streamlit/secrets.toml*

private\_gsheets\_url = "{link to Google Sheet}"

[gcp\_service\_account]

type = "service\_account"

project\_id = "xxx"

*# … rest of the key file*

**IMPORTANT:** The just created .streamlit/secrets.toml file should be added to the gitignore file in the repository. Otherwise, the Google Sheet URL and information in the secrets file will become public if the code is pushed to Github.

### Upload Secrets to Steamlit Cloud

Because the .streamlit/secrets.toml is not pushed to Github, Streamlit does not have access to the Google Sheet URL and secrets information yet. Please copy and paste the information in the .streamlit/secrets.toml here: [Streamlit app overview](https://share.streamlit.io/)[[11]](#footnote-11) → 3 dots next to the app → Settings → Secrets.

### Store User Input to Sheet

At this point, all the connections are working. There are some last steps that are needed to be able to write user input to the Google Sheet.

1. Import the following in the Python script:

import gspread

from google.oauth2 import service\_account

1. Add gspread=={version} to requirements.txt in the repository to let Streamlit Cloud know to use this package.
2. Set up the scope and credentials to the Google Sheet:

credentials = service\_account.Credentials.from\_service\_account\_info(

st.secrets["gcp\_service\_account"],

scopes=[

"https://www.googleapis.com/auth/spreadsheets",

],

)

1. Open the Google Sheet as sh:

client = gspread.authorize(credentials)

sheets\_url = st.secrets["sheet\_url"]

sh = client.open\_by\_url(sheets\_url)

1. Write user input as a row to the Google Sheet:

sh.sheet1.append\_row([user\_input])

If everything went correctly, the user input will be visible in the Google Sheet. Appendix A provides an overview of the complete example code. Appendix B provides an overview of the repository structure.

TODO more about actual current dashboard? E.g., screenshot of UI, explanation about weights selection etc

# Risk Score Calculation

The dashboard lets the user score risk drivers, which are then weighted by expert weights and converted to a single risk score.

## Scoring of Risk Drivers

The model includes 6 risk drivers [TODO benoemen]. Each risk driver can consist out of a maximum of 4 variables. As an example, the Risk Driver *Resource availability* consists of the following variables:

1. Dependency on critical raw materials.
2. Closest peak year of critical raw materials.
3. Ownership/control over resources (natural hedge).
4. Type of relationship with value chain.

Each of these variables is scored out of four scores [TODO checken]. 1 is a ‘bad’ score, indicating a risk. 4 is a ‘good’ score, indicating little risk.

For each risk driver, a score is calculated based on the average of its variables. This score is inverted, so that it ranges from 0.25 (good) to 1 (bad). Then, this range is converted so that it starts at 0 (good), up to 1 (bad).

## Including Expert Weights

When each risk driver is scored, they need to be weighted by the corresponding expert weights. These weights are an average of the chosen expert groups, which are:

1. Risk
2. Bus Dev (Business Development)
3. Finance
4. Invest (Investors)

Note that the *Total avg score* is not provided as an option. This score is weighted on the size of the expert groups, and therefore biased to large expert groups.

The risk driver score is multiplied by the average of the chosen weights corresponding to this risk driver. Afterwards, the average of the weighted risk driver scores is calculated, by dividing the sum of weighted risk drivers by the amount of risk drivers.

TODO expert weight normalisation uitleggen

# Concerns

This chapter discusses concerns related to the model.

## Access to dashboard on Github:

The current dashboard is hosted on [my Github page](https://github.com/irisreitsmaRQ/dasbhoard)[[12]](#footnote-12), which is connected to my RiskQuest email. From September, I will not work at RiskQuest anymore. My email might be deleted, so the Github might not be accessible anymore.

TODO more?

Appendix

1. Complete Example Code

This appendix provides an example of a simple dashboard code in Python that is stored in a Google Sheet.

# *simple\_script.py*

import streamlit as st

import gspread

from google.oauth2 import service\_account

# Create simple dashboard for user input

st.write(‘Hello world!’)

user\_input = st.text\_input(label=’Say something’, value=’Something’)

# Set up credentials to Google Sheet

credentials = service\_account.Credentials.from\_service\_account\_info(

st.secrets["gcp\_service\_account"],

scopes=[

"https://www.googleapis.com/auth/spreadsheets",

],

)

# Authorize and open and Google Sheet

client = gspread.authorize(credentials)

sheets\_url = st.secrets["sheet\_url"]

sh = client.open\_by\_url(sheets\_url)

# Store user input in Google Sheet

sh.sheet1.append\_row([user\_input])

1. Repository Structure

The complete repository structure for the example code should look like:

* Repository
  + .streamlit
    - Secrets.toml
  + Simple\_script.py
  + Requirements.txt
  + .gitignore

1. <https://dasbhoard-7sgf0oydm6l.streamlit.app/> [↑](#footnote-ref-1)
2. <https://github.com/irisreitsmaRQ/dasbhoard> [↑](#footnote-ref-2)
3. <https://streamlit.io/> [↑](#footnote-ref-3)
4. <https://github.com/> [↑](#footnote-ref-4)
5. <https://www.google.com/sheets/about/> [↑](#footnote-ref-5)
6. <https://streamlit.io/cloud> [↑](#footnote-ref-6)
7. <https://docs.streamlit.io/library/get-started/installation> [↑](#footnote-ref-7)
8. <https://blog.streamlit.io/host-your-streamlit-app-for-free/> [↑](#footnote-ref-8)
9. <https://docs.streamlit.io/library/get-started/create-an-app> [↑](#footnote-ref-9)
10. <https://docs.streamlit.io/knowledge-base/tutorials/databases/private-gsheet> [↑](#footnote-ref-10)
11. <https://share.streamlit.io/> [↑](#footnote-ref-11)
12. <https://github.com/irisreitsmaRQ/dasbhoard> [↑](#footnote-ref-12)