

The **Polya's Shiny Urn** R-Package

Mitja Seibold - 11383232

Department of Psychology, University of Amsterdam, the Netherlands

Abstract

This document explains the usage of the Polya's Shiny Urn R-Package. This R-Package includes the Clinical Trial Dilemma, a theoretical model, as a Polya's Urn Model Simulation incorporated into a Shiny App. The clinical trial dilemma explains the problem that at a clinical trial on the one hand you want to make sure that the treatments tested are randomized. On the other hand, you want to make sure that most of the patients will receive the best treatment possible. A sequential Polya's Urn can simulate this. With this shiny app you have the option of a simple and an advanced mode where different inputs can generate different simulation scenarios.

This step by step guide helps you to install the package and shows you how to use it. In addition it gives you background informations. At the end an example is presented.

Keywords: Polya's Urn Model, Clinical Trial Dilemma, Shiny App

SHORT INTRODUCTION

This Shiny App represents the clinical trial dilemma - where you, as a doctor, wants to conduct a sequential clinical trial to see which treatment is best. While on the one hand making sure that the treatments get tested randomly, on the other hand you also want to make sure that most patients get the best treatment possible. This can be represented as a Polya's Urn, where the urn includes balls of different colors (each treatment represents one color) and each patient gets one draw out of the urn - so randomly gets one treatment assigned to. If the treatment was successful one or more (depending on the condition) additional ball(s) of the same treatment will be added to the urn. Now, for the next patient again a treatment will be drawn out of the urn, however the ratio of the balls (the ratio between treatments) has changed so that there a more balls of successful treatments in the urn than treatments that were unsuccessful.

INSTALLATION

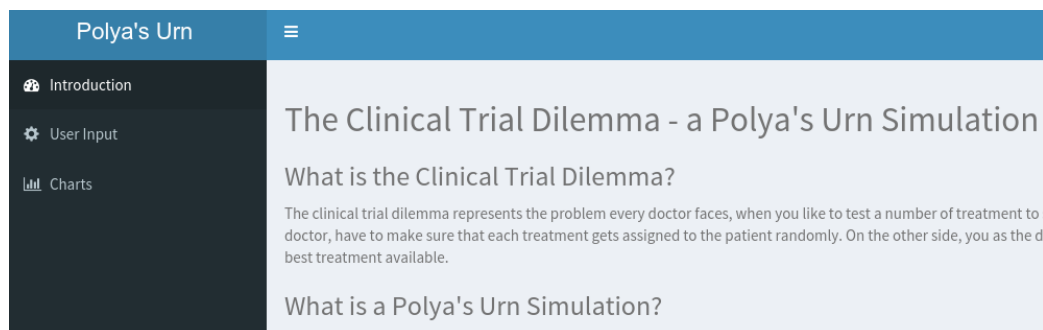
[CAUTION]: To be able to use the Shiny app R Studio ¹, Shiny² as well as Shiny Dashboard³ needs to be installed! For easier installation process the `devtools` package should also be installed.

To install and be able to use the package do the following:

1. Use the command line in R and type `install_github("81N55E/PNS_Polya")`⁴ to download and install the package in one go. Otherwise download the `PolyasShinyUrn.tar.gz` package file from Github into your preferred working directory & then
2. (if you have not done it in step 1) - set your working directory where your file was downloaded to and install package use your terminal and type in R CMD `PolyasShinyUrn.tar.gz` where the file is located.
3. In R type `library(PolyasShinyUrn)` to load the functions into your workspace.

HOW TO USE THE APP

Starting the Shiny App. To start the app you have to run the `polyasShinyRoundup()` - function. When the app is initiated a new window has popped up that should look like this:



On the left side of the app the Dashboard with three options can be seen. The first option is the introduction panel that is always shown first when the program is started. The second option is the input page, where the user can make all the necessary inputs for the simulation to run and the third panel is the charts panel where the charts and additional information is presented depending on the input given.

¹Rstudio

²`install.packages("shiny")`

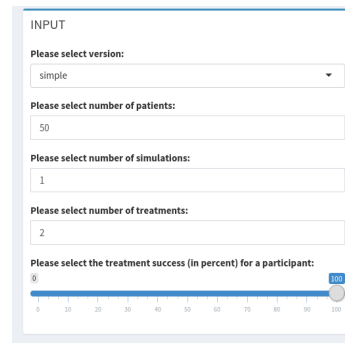
³`install.packages("shinydashboard")`

⁴ `install_github()` is a function in the `devtools` Package.

The Introduction Panel. The first panel that you see after starting the program is the Introduction Board. Here you can get a quick description of the tasks background and how to use the app in detail as well as some FAQs.

The User Input Panel. The second panel is the user input panel where you can (not surprisingly) declare all the necessary criteria for your simulation. There is the option to choose between two versions - a simple version and an advanced version.

The Simple Version looks like this:



The 'INPUT' panel for the simple version contains the following fields:

- Please select version:** A dropdown menu with 'simple' selected.
- Please select number of patients:** A text input field with '50' entered.
- Please select number of simulations:** A text input field with '1' entered.
- Please select number of treatments:** A text input field with '2' entered.
- Please select the treatment success (in percent) for a participant:** A horizontal slider ranging from 0 to 100, with the value set to 100.

The options are as follows:

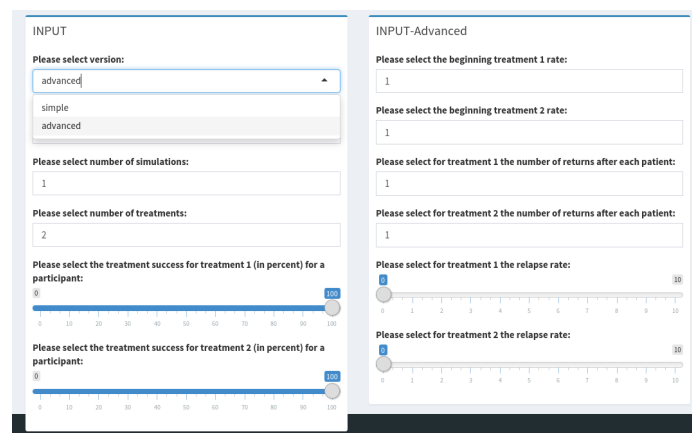
Patients: Here you can choose the number of patients you want to test. You test one patient after another. *Hint: The patients represent the number of trials.*

Simulations: Here you can choose the number of simulations - that is how often do you want to do the experiment.

Treatment: Here you can choose the number of treatments you want to test. *Hint: You want to test at least two treatments.*

Success Rate: Here you can choose the success rate of the treatments - that is a probability how successful any treatment is for each patient (0-100%).

The Advanced Version looks like this:



The 'INPUT-Advanced' panel contains the following fields:

- Please select version:** A dropdown menu with 'advanced' selected.
- Please select number of simulations:** A text input field with '1' entered.
- Please select number of treatments:** A text input field with '2' entered.
- Please select the treatment success for treatment 1 (in percent) for a participant:** A horizontal slider ranging from 0 to 100, with the value set to 100.
- Please select the treatment success for treatment 2 (in percent) for a participant:** A horizontal slider ranging from 0 to 100, with the value set to 100.
- Please select the beginning treatment 1 rate:** A text input field with '1' entered.
- Please select the beginning treatment 2 rate:** A text input field with '1' entered.
- Please select for treatment 1 the number of returns after each patient:** A text input field with '1' entered.
- Please select for treatment 2 the number of returns after each patient:** A text input field with '1' entered.
- Please select for treatment 1 the relapse rate:** A horizontal slider ranging from 0 to 10, with the value set to 0.
- Please select for treatment 2 the relapse rate:** A horizontal slider ranging from 0 to 10, with the value set to 0.

The options for the advanced version are as follows:

Patients: Same as in the simple version.

Simulations: Same as in the simple version.

Treatments: Same as in the simple version.

Success Rate: Here you can choose the success rate for the first two treatments individually.

Start Rate: Here you can choose the start rate for the first two treatments individually.

Return Rate: Here you can choose the return rate for the first two treatments individually.

Relapse Rate: Here you can choose the relapse rate for the first two treatments individually.

The Charts Panel. The chart panel is the panel where you can find the output after the simulation. Depending on your input, the output changes accordingly. Additionally you have the option to change which and how many treatments should be plotted (Although the maximum number of treatments that are plottable are 3). The charts panel looks as follows:



One top left panel you can see the number of treatments that you chose. On the middle top panel you can see the number of simulations that you chose. And on the right top panel you can see the number of patients that you chose in the input panel. The graph below represents the ratio of the chosen treatments to be plotted. That is the y-axis represents the ratio of each treatment and the x-axis represents the number of treatments. If more than one simulation is chosen lines of the same color represent different simulations. In default mode the red line represents the first (not necessarily the best) treatment, while the blue line represents the second treatment in default mode. you can change which treatment should be plotted (see below). On the right side beside the plot three panels represent the best three treatments over all simulations. The fourth panel (in light blue) represents the percentage of how often the best treatment was actually the best after all simulations). Of note, if only two treatments were chosen for the simulation (obviously) only the best two treatments are indicated (and the third panel will no be visible but instead a red error message will appear - that you should not be bothered with).

Below the graph you can find four additional input panels.



In the three panels right below the graph you can indicate which treatment should be plotted by which color. In the panel below you can use the slider to indicate how many treatments should be plotted.

BACKGROUND

The Clinical Trial Dilemma. The clinical trial dilemma represents the problem every doctor faces, when you like to test a number of treatment to see which works best. On the one side, you as the doctor, have to make sure that each treatment gets assigned to the patient randomly so that there is no bias influencing your result and your groups representing the population you are testing are equal. On the other side, you as the doctor, want to make sure that each patient gets the best treatment available. Thus, testing multiple treatments for which works best and at the same time treating your test subjects with the best treatment possible causes a dilemma.

Polya's Urn. Traditionally Polya's Urn represents an urn with two balls of different color (e.g., one black, one white). At each trial one ball is drawn. Of the ball drawn the color is observed (e.g., white). The ball is then returned into the urn and an additional ball of the same color is added into the urn. Then the next trial starts, where again a ball is drawn randomly. The Polya's Urn model has been used in cognitive science to for example simulate mental exercise (e.g. learning) or in bio science (e.g., genetics and cell division) but has also been used in social science to for example model "the rich get richer and the poor get poorer" paradigm. The parameters of the model can be changed such as different numbers of balls in the beginning, adding more colors, or change of return rate. See Wikipedia for more details.

Clinical Trial Dilemma and Polya's Urn. In a mathematical theory paper published in 1979 by Wei⁵, he indicated that the clinical trial dilemma can be modelled by a Polya's urn. This Polya's urn represents a sequential medical trial where treatments get randomly assigned to patients, yet successful treatments get more assigned to patients than not so successful treatments. Therefore, the different treatment are represented by different colored balls. The number of patients represent the number of trials (or draws out of the urn), while the one urn represents one simulation. Thus, in a sequential matter, a patient

⁵ Wei, L. J. (1979). The generalized Polya's urn design for sequential medical trials. *The Annals of Statistics*, 291-296.

comes to the doctor and the doctor draws one ball (the specific treatment) out of an imaginary urn. After the patient is treated the success rate accounts whether an additional ball is added.

In this Shiny app, additional features are incorporated that might represent real live scenarios. For example can you assign different success rates to two treatments if you know already that you expect the one treatment to perform better over the other. In addition you can change the return rate, indicating that if the one treatment is successful this should be supported by returning more than one ball back into the urn. The relapse rate indicates that although a patient was treated successfully with a specific treatment there is a certain percentage of patients that will relapse - thus they will get sick again, which indicate that the treatment might not be as good in the long run.

DESIGN

The Flowchart in Figure 1 below explains the program design with the flow of the internal calculation in more detail. On the left side is the user interface, that is all the input that comes from the user. On the right side is the internal program and how the user interface affects the outcome of the program. On the bottom the outcome is visualized in the shiny app.

EXAMPLE

These are two short examples how the shiny app can be used. First is an example for the simple mode. The second example shows how the advanced mode might be used.

Simple Mode Example. Let us assume we are a doctor who wants to test three new treatments for anxiety to see which one works best. There has been studies done on rats and they seem promising with a 70% chance that the treatments are successful in an individual. Interestingly all three new treatments seem to work equally well. So our job is now to test these three treatments on normal patients with anxiety disorder. As we have some knowledge about correct clinical trials we know we have to randomize our trials so that there is no bias affecting our results. So each patient will be randomly assigned to a treatment. On the other hand, we are a clinician and want to make sure that most patients treated will receive the best treatment possible. We ran a power analysis and came up with a total sample size of 75 to see an effect. However, before we want to start the real clinical trial we want to run a simulation to see what we can expect. Luckily there is a Shiny App that does exactly that.

After starting the app we can use the user interface to set our parameters (see Figure 2). The number of patients is set to 75. The number of simulations is set to 1. The number of treatments are set to 3 and the success rate is set to 70% because of the previous animal research. Then we click the "Chart"-Button on the Dashboard and the results pop up.

What does the output tells us? On the top we can make sure that our input was correct. On the right side panel we see, that the best treatment is treatment onw. Because we have only one simulation the rate is at 100%. On the graph only the first two treatments are plotted. Therefore, we have to scroll down to change the input so it looks like this

Figure 1. This is the flowchart that indicates the programming design behind this shiny app. The different colors represent loops within the the functions

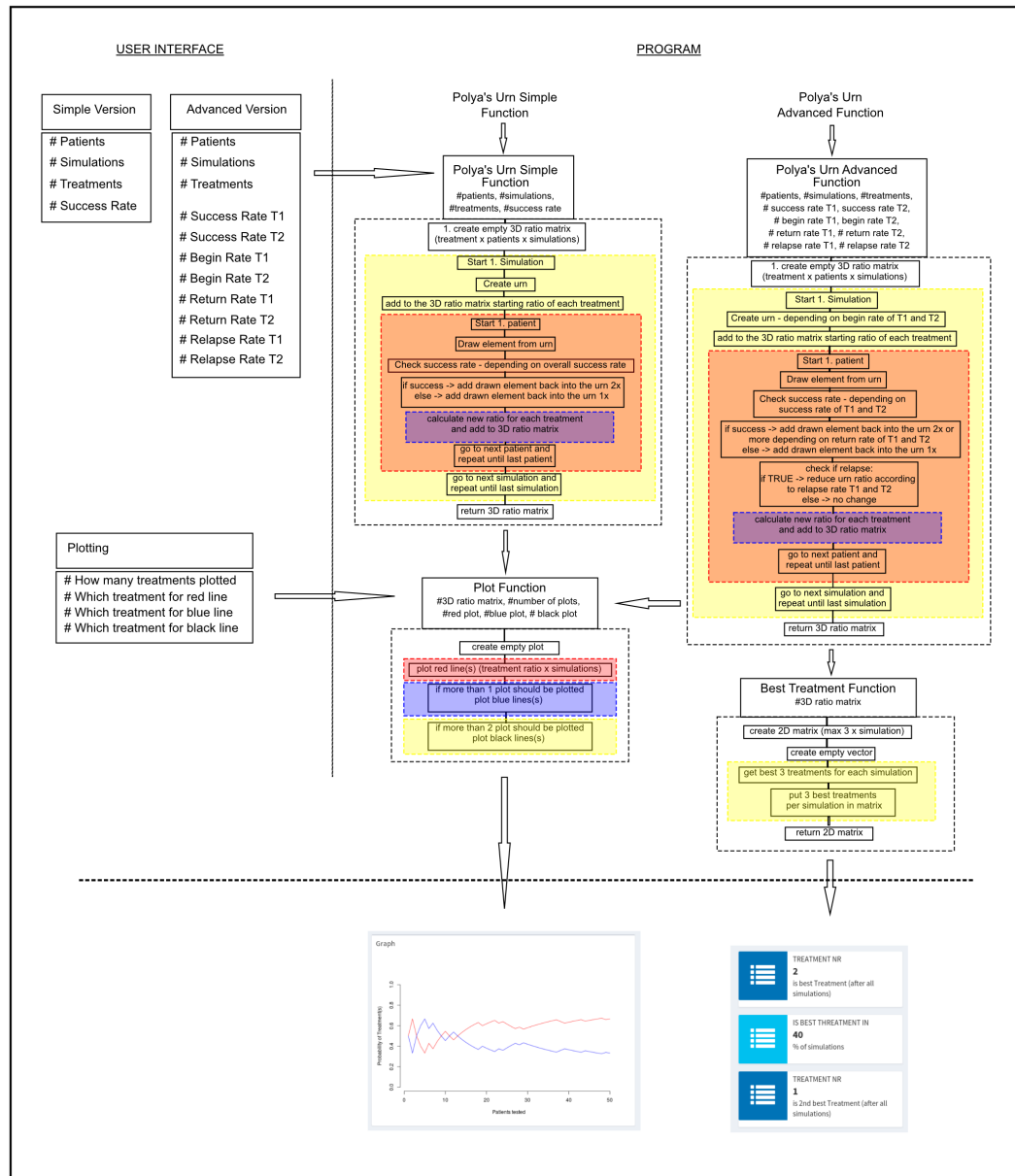


Figure 2. Input for Simple Example

INPUT

Please select version:
simple

Please select number of patients:
75

Please select number of simulations:
1

Please select number of treatments:
3

Please select the treatment success (in percent) for a participant:
0 70 100

- add another line to the plot (we don't have to change the plot treatments as they are already in the right order):

Patients tested

simulations)

Which Treatment should be plotted in Red?
1

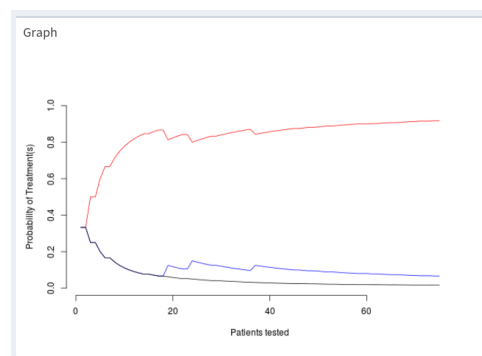
Which Treatment should be plotted in Blue?
2

Which Treatment should be plotted in Black?
3

Number of Treatments Plotted

Please select number of best treatments plotted (max 3):
1

As we can see now, the plot has changed - including one new black line with the lines representing different treatments. First of all it seems like that it is very clear that treatment 1 seems to be the best one with treatment 2 and 3 with a very low ratio.

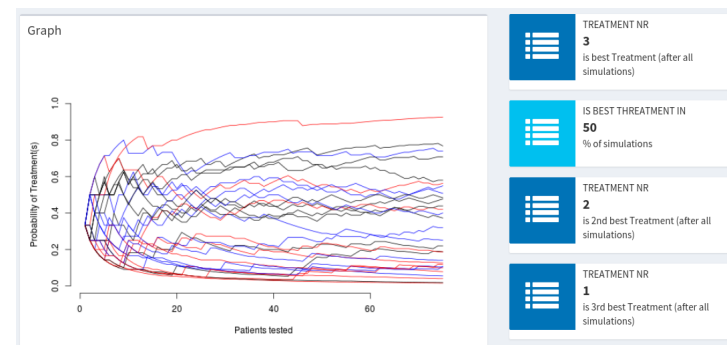


Therefore, it seems easy to choose which treatment is superior over the others. However, this should be treated with caution. An option to make really sure if treatment 1 is superior

Figure 3. Output for Simple Example - [CAUTION: This might look very different from your output]



over the other two is to run more than one simulation. So let's update the input and let's assume instead of one experiment we would do 10 individual experiments with 75 patients each. How would that change our outcome?



As we can see, after 10 simulations, the best treatment has changed. Now, treatment 3 is the best treatment - in 50% of the simulations. The first simulation looked like an outlier. However, as this is a random simulation this result should be treated with caution. If we would do the same simulation again there is a high chance of a different result.

Advanced Mode Example. For the Advanced Example we picture ourselves in a psychiatry. We have several patients with depression and have two options for a treatment - first is basic cognitive behavioural therapy (CBT) and the second is Depth-Psychology therapy (DPT). We want to test 50 patients and randomly assign one of the two treatments to them. In our example each patient gets 8 weeks of therapy. We plan to test each patient after another. We know from previous research that the success rate of CPT is better than DPT - with a rate of around 70:40. The relapse rate, however, is greater with CPT than DPT - with a rate around 70:20. The begin rate and return rate is equal with both therapies. Although we can expect that CPT will perform better, the high relapse rate makes us unsure if this CPT really outperforms DPT. The input would look like this:

INPUT

Please select version:
advanced

Please select number of patients:
50

Please select number of simulations:
1

Please select number of treatments:
2

Please select the treatment success for treatment 1 (in percent) for a participant:
75

Please select the treatment success for treatment 2 (in percent) for a participant:
40

INPUT-Advanced

Please select the beginning treatment 1 rate:
1

Please select the beginning treatment 2 rate:
1

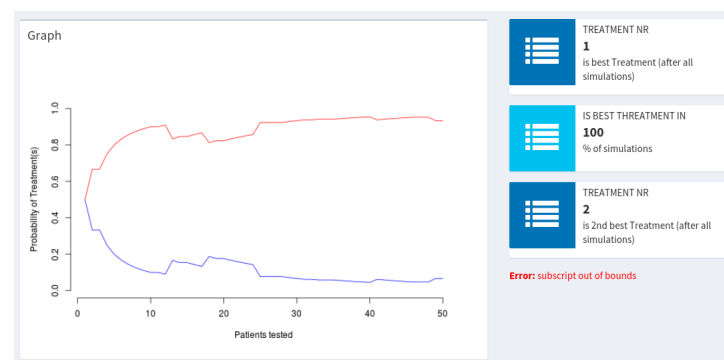
Please select for treatment 1 the number of returns after each patient:
1

Please select for treatment 2 the number of returns after each patient:
1

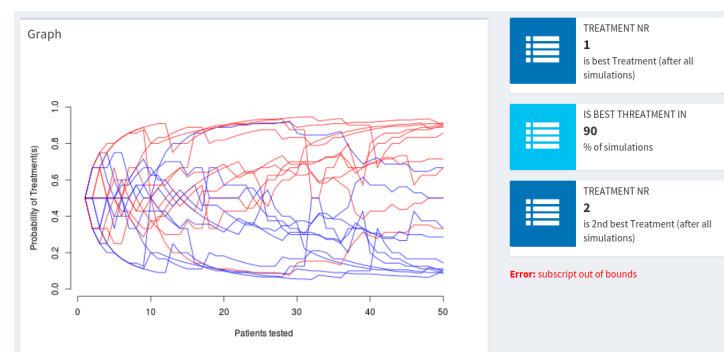
Please select for treatment 1 the relapse rate:
7

Please select for treatment 2 the relapse rate:
2

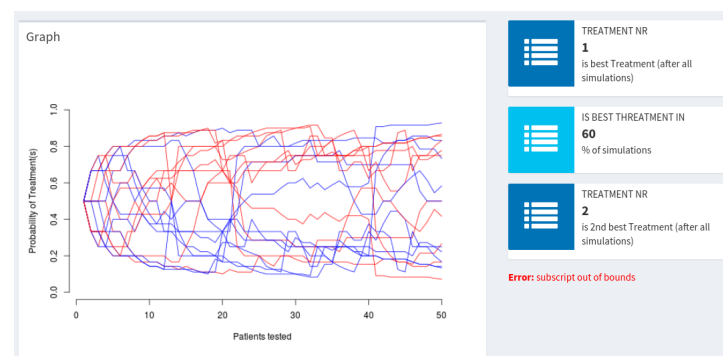
The output panel would the look like this:



As we can see, even with the high relapse rate, CPT seems to outperform DPT. To make sure that this is not a coincidence we should run our simulation multiple time - as in the simple example, let us say 10 times. Does that change our findings?



As the updated output indicates that the best treatment seems to be CPT - even after several simulations. As it is the case in 90% of the simulation we can assume that this is a very solid finding. But how would the result change if CPT has not such a high success rate as expected, that is rather a success rate of 60 instead of 70?



The updated simulation indicates that the CPT is still superior, yet it dropped from 90% to 60% in its success over all treatments which can be interpreted as a big step.

VERSION CONTROL

Version 0.0 - week one.

- Finding the topic.
- Creating the flowchart draft and brainstorming ideas. First pseudo-code draft of the setup.

Version 0.1 - week two.

- Programming simple polya's urn in R.
- Creating first Shiny App with simple (patients, simulations and treatments = 2) and intermediate option (number treatments can be changed by the user).
- The output in the simple mode is one graph with the first two treatments and in the intermediate mode a plot of the best two treatments. Each mode has an individual polya's urn function including the plotting.
- The advanced mode is still a thought concept.

Version 0.2 - week three.

- For the Shiny App a Dashboard is included with Introduction Panel, Input Panel and Output Panel.
- The Advanced mode is created including for the first two treatments individual number of starting rate, individual number of return rate and individual number of relapse rate.
- All function are changed greatly to incorporate the reactive() function for shiny. This included that all individual *mode* functions are merged into one function. The plotting is outsourced into an individual function.
- Additional output panels are incorporated into Shiny and presented besides the output-plot to indicate the best 3 treatments over all simulations as well as the basic setup chosen.

Version 0.3 - week four.

- The simple and intermediate mode is merged into the *simple* mode.
- The Success rate is programmed for simple (success rate for all treatments the same) as well as advanced mode (for first and second treatment success rate can be individualized).
- The plotting function is changed so now the user can individually change the treatments plotted.
- a new box was included in the output panel to indicate winning percentage.
- The `bestTrt()` function was changed to return only the matrix (and not a vector).
- The function file (that incorporated all necessary function) is changed into individual function files (for each function one file).
- Introduction panel for Shiny App is written.
- `ui.R` file and `server.R` file are included into one function that that the whole app can be called from the console.
- Documentation is written.
- R-Package is created including help-function etc.