jsPsychR manual

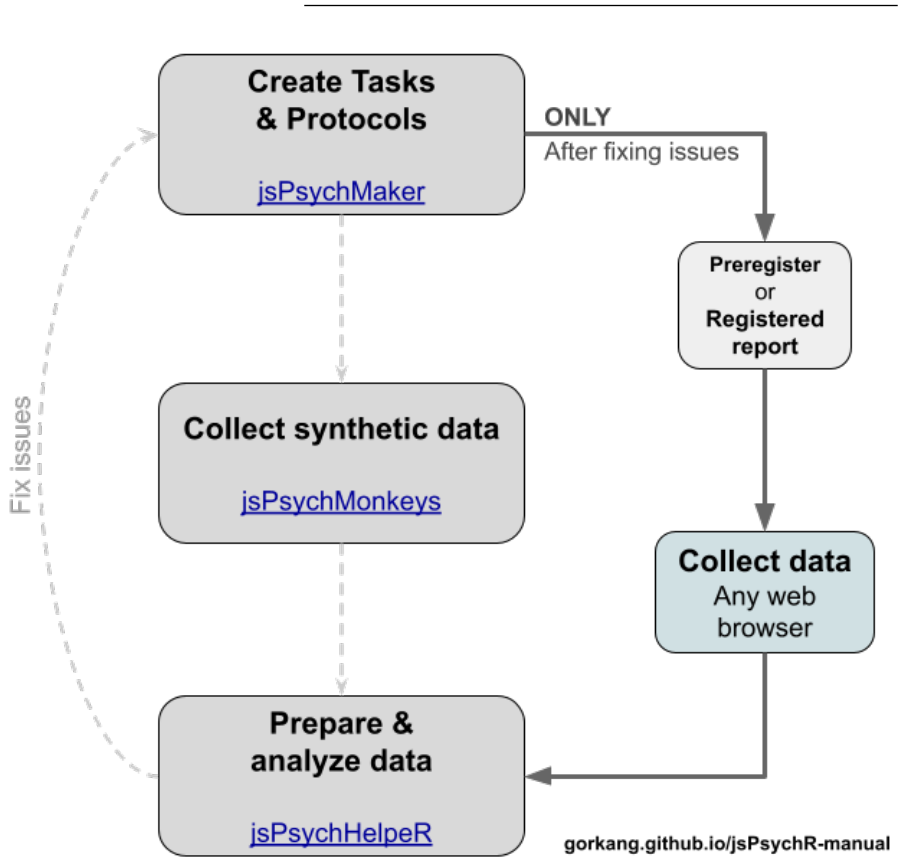
Gorka Navarrete

2023-08-21

Table of contents

What is jsPsychR?

`jsPsychR` is a group of **open source** tools to help create experimental paradigms with jsPsych, simulate participants and standardize the data preparation and analysis. The final goal is to help you have the data preparation and analysis ready before collecting any real data, drastically reducing errors in your protocols, and making the move towards registered reports easier.



We have three main tools:

- jsPsychMaker: Create experiments with jsPsych, randomize participants, balance between conditions, reuse already existing tasks, etc.
- jsPsychMonkeys: Release monkeys to a jsPsych experiment with {targets}, docker and {RSelenium}
- jsPsychHelpeR: Standardize and automatize data preparation, analysis and reporting of jsPsych experiments created with jsPsychMaker

And a package for the system admins:

- jsPsychAdmin: Functions to help with common administrative tasks for jsPsychR protocols

---

Contributors

- Gorka Navarrete
- Herman Valencia

Papers published using jsPsychR

- Neely-Prado, A., van Elk, M., Navarrete, G., Hola, F., & Huepe, D. (2021). Social Adaptation in Context: The Differential Role of Religiosity and Self-Esteem in Vulnerable vs. Non-vulnerable Populations–A Registered Report Study. Frontiers in Psychology, 12, 5257. https://doi.org/10.3389/fpsyg.2021.519623

1 Reproducible experiments

We use different technologies to develop experiments. Some examples are Psychopy, Qualtrics, Limesurvey, jsPsych, Gorilla, etc. Each of these has advantages and disadvantages and, in general, there are pragmatic aspects to take into account when adopting one or the other: cost, type of experiment (EEG or behavioral, lab or online), lab history and available resources, ...

We opted for jsPsych to run behavioral experiments because it is an **open source** javascript library, based on standard web technologies, and can be used online and offline.

In the last years, we started working on a set of tools to help people without coding expertise to create jsPsych experiments (jsPsychMaker), simulate participants (jspsychMonkeys) and standardize and automatize the data preparation and analysis (jsPsychHelpeR).

Our final goal is to have a big catalog of tasks available to use in the jsPsychMaker repo. Each of the tasks should run with jspsychMonkeys to create virtual participants. And each task will have a sister script in jsPsychHelpeR to fully automate data preparation.

1.1 Open and reproducible pipeline

To replicate an experiment from a publication is not trivial. One of the main goals of this sytem is to be able to create, share and reproduce an experiment, its data, and data preparation and analysis without any extra effort.

Furthermore, all the components of the pipeline must be Open Source, which allows reviewers, collaborators, etc. to check and run the code. This also makes it accessible to anyone with a computer and access to the internet, eliminating cost constrains.

With this system you can create a paradigm, simulate data and prepare data and analysis almost automatically.

The system output is standardized, so names of variables and the structure of the data are predictable. Finally, the plots, tables, reports and analysis are reproducible, so you can get everything ready with simulated data, preregister or even better, go for a registered report and just relaunch the data preparation and analysis when the participant's data arrive, with a single command.

And if you want to share the final data preparation and analysis project in a Docker container to make sure the future generations will be able to run it without dependency issues, we got you covered.

1.2 Automatization

We tried to get a few basic things right, but this is an evolving project, and some things are more complex than one would want. Please do report the issues you find:

- jsPsychMaker issues
- jsPsychMonkeys issues
- jsPsychHelpeR issues

Figure 1.1: SOURCE: https://xkcd.com/1425/

2 Quick Guide

Pre-requisites

1. Install R
2. Install RStudio desktop
3. Install Docker. See the Monkeys' setup section for more detailed instructions.

2.1 jsPsychMaker: Create an experimental protocol

See the jsPsychMaker chapter for more detailed instructions.

Outline

1) Install jsPsychMaker
2) `create_protocol()` using any of the `list_available_tasks()` and edit the `config.js` to adapt the protocol settings
3) Open `index.html` in your browser

---

1) Install jsPsychMaker

Open RStudio and run the following line in the console. This will install the jsPsychMaker package from the Github repository.

```
if (!require('remotes')) install.packages('remotes'); remotes::install_github("gorkang/jsP

# If you are on Ubuntu and you get an igraph error, try: sudo apt install build-essential
```

2) Create protocol

Create and test a fully working protocol with `jsPsychMaker::create_protocol()`.

Include the `canonical_tasks` you want (list the available tasks with `jsPsychMaker::list_available_tasks()`)
You have more details in available-tasks. If you need new tasks, see New tasks.

```
jsPsychMaker::create_protocol(canonical_tasks = c("AIM", "EAR", "IRI"),
                              folder_output = "~/Downloads/protocol999",
                              launch_browser = TRUE)
```

You must edit `config.js` to adapt the protocol to your needs. See experiment configuration for more details.

3) Run experiment

The experiment is ready to run on your computer. Open `index.html` in Google Chrome or your favorite (and up to date) browser.

2.2 jsPsychMonkeys: Simulate participants

See the jsPsychMonkeys chapter for more detailed instructions.

---

jsPsychMonkeys uses Selenium inside a Docker container to guarantee each session is a clean session. See how to setup your computer.

Outline

1) Install jsPsychMonkeys

2) Run Monkeys

---

1) Install jsPsychMonkeys

```
if (!require('remotes')) utils::install.packages('remotes'); remotes::install_github('gork

# If you are on Ubuntu and you get an igraph error, try: sudo apt install build-essential
```

2) Run Monkeys

Use the `uid` parameter to set the participants' numeric id's, e.g. `uid = 1:10` would launch monkeys 1 to 10.

Use the `local_folder_tasks` parameter to indicate the location of the `jsPsychMakeR` protocol. If you are on Windows, `local_folder_tasks` value should be something similar to `C:/Users/myusername/Downloads/protocol999`.

```
jsPsychMonkeys::release_the_monkeys(uid = 1:10,
                                    local_folder_tasks = "~/Downloads/protocol999/")
```

If the protocol was running from a local folder, the Monkey's responses will be copied to a subfolder `.data/` inside the `local_folder_tasks`. In the example above, `~/Downloads/protocol999/.data`. If the protocol was running on the server (see the `server_folder_tasks` parameter), the data will be in the protocols' `.data/` folder inside the server.

2.3 jsPsychHelpeR: Prepare data

See the jsPsychHelpeR chapter for more detailed instructions.

Outline

1) Install jsPsychHelpeR
2) Create new project
3) Run data preparation

---

1) Install jsPsychHelpeR

- Install jsPsychHelpeR from Github.

```
if (!require('remotes')) utils::install.packages('remotes'); remotes::install_github('gork

# If you are on Ubuntu and you get an igraph error, try: sudo apt install build-essential
```

2) Create new project

Create and setup a new RStudio project for your data. Before doing this, you need to locate the raw data for the jsPsychMaker project.

In this example, our raw data is in `~/Downloads/protocol999/.data/` and we want the new project to be in `~/Downloads/jsPsychHelpeR999/`

```
jsPsychHelpeR::run_initial_setup(pid = '999',
                                 data_location = '~/Downloads/protocol999/.data/',
                                 folder = '~/Downloads/jsPsychHelpeR999/')
```

After this, **a new RStudio project will open**.

3) Run data preparation

Run the data preparation **in the new RStudio project** with `targets::tar_make()`

```
# Restore all the necessary packages using renv
renv::restore(prompt = FALSE)

# Run data preparation
```

```
targets::tar_make()
```

If you are curious, running `targets::tar_visnetwork(targets_only = TRUE)` will show the whole data preparation targets tree. Open the file `run.R` for more details.

---

3 jsPsychMaker

---

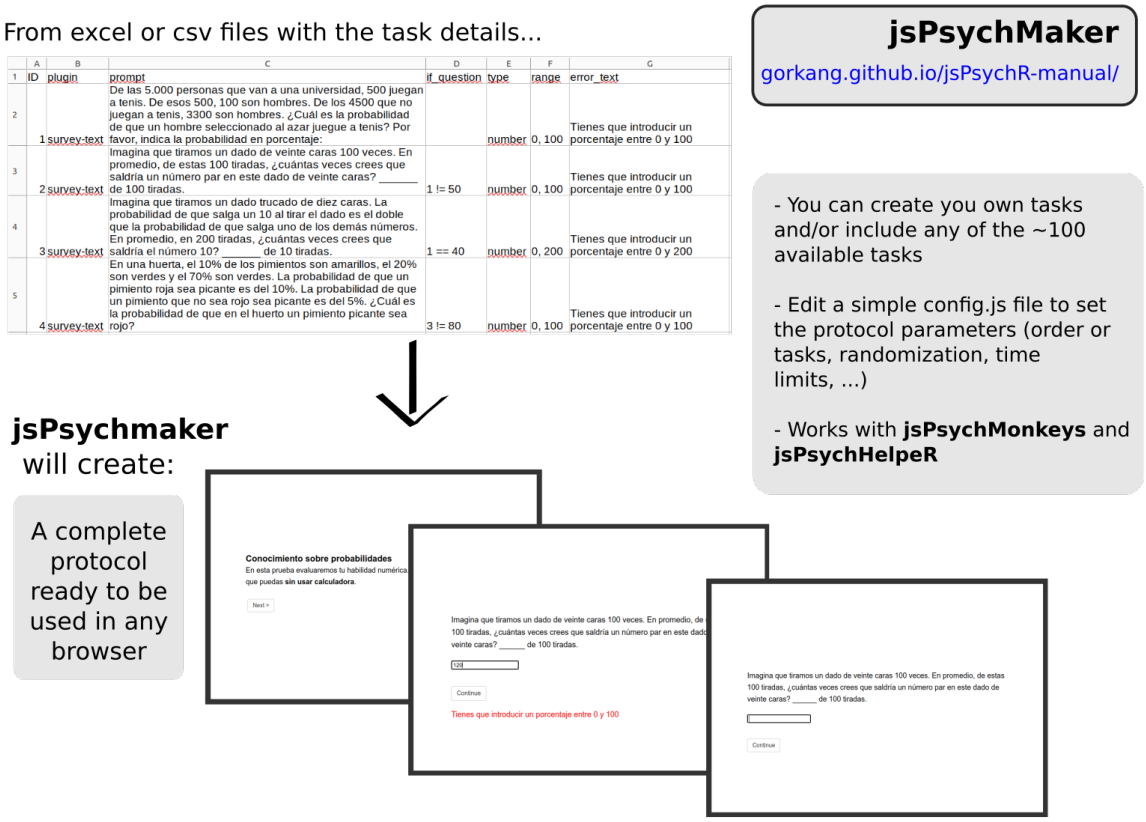
jsPsychMaker: Create experiments with jsPsych, randomize participants, etc.

In brief

Using jsPsychMaker to build experimental protocols helps you with a few things:

- Create full protocols using:
  - tasks already implemented
  - task that will be created reading a csv or excel file
- Configure your protocol by editing a simple config.js file
  - You can also use the jsPsychMaker Shiny APP to edit your config.js file
- Select order of tasks, randomize blocks of tasks, etc.
- Randomize participants to groups, making sure the balance between the groups is maintained
- Allow participants to continue in the task where they left in the protocol
- Set time limits to complete the protocol
  - Automatically discard participants over the time limit, freeing the slots for new participants
- Run online and offline protocols
- Simulate participants with jsPsychMonkeys
- *Automagically* get your data prepared with jsPsychHelpeR

See QuickGuide for basic instructions.



	ID	plugin	prompt	if_question	type	range	error_text
1	1	survey-text	De las 5.000 personas que van a una universidad, 500 juegan a tenis. De esos 500, 100 son hombres. De los 4500 que no juegan a tenis, 3300 son hombres. ¿Cuál es la probabilidad de que un hombre seleccionado al azar juegue a tenis? Por favor, indica la probabilidad en porcentaje:		number	0, 100	Tienes que introducir un porcentaje entre 0 y 100
2	2	survey-text	Imagina que tiramos un dado de veinte caras 100 veces. En promedio, de estas 100 tiradas, ¿cuántas veces crees que saldría un número par en este dado de veinte caras? _____ de 100 tiradas.	1 != 50	number	0, 100	Tienes que introducir un porcentaje entre 0 y 100
3	3	survey-text	Imagina que tiramos un dado trucado de diez caras. La probabilidad de que salga un 10 al tirar el dado es el doble que la probabilidad de que salga uno de los demás números. En promedio, en 200 tiradas, ¿cuántas veces crees que saldría el número 10? _____ de 10 tiradas.	1 == 40	number	0, 200	Tienes que introducir un porcentaje entre 0 y 200
4	4	survey-text	En una huerta, el 10% de los pimientos son amarillos, el 20% son verdes y el 70% son verdes. La probabilidad de que un pimiento roja sea picante es del 10%. La probabilidad de que un pimiento que no sea rojo sea picante es del 5%. ¿Cuál es la probabilidad de que en el huerto un pimiento picante sea rojo?	3 != 80	number	0, 100	Tienes que introducir un porcentaje entre 0 y 100

3.1 Available tasks

In 2023-08-21 we have 69 tasks implemented, and 21 in development. The full details about the available tasks can be checked in this document. You can always check the full list of tasks in the Github repo.

To list the available tasks in you console, you can use the `list_available_tasks()` function. If you don't have the `jsPsychMaker` package, install it first.

```
jsPsychMaker::list_available_tasks()$tasks
```

```
 [1] "AIM"          "Bank"         "BART"         "BNT"          "bRCOPE"
 [6] "CAS"          "Consent"      "ConsentHTML"  "Cov19Q"       "COVIDCONTROL"
[11] "CRS"          "CRT7"         "CRTMCQ4"      "CRTv"         "DASS21"
[16] "DEBRIEF"      "DEMOGR"       "EAR"          "EmpaTom"      "ERQ"
[21] "ESM"          "fauxPasEv"    "GBS"          "GHQ12"        "Goodbye"
[26] "HRPVB"        "HRPVBpost"    "IBT"          "IDQ"          "IEC"
[31] "INFCONS"      "IRI"          "IRS"          "MDDF"         "MDMQ"
[36] "MIS"          "OBJNUM"       "OTRASRELIG"   "PBS"          "PRFBM"
[41] "PRFBMpost"    "PSETPP"       "PSPPC"        "PSS"          "PVC"
[46] "PWb"          "REI40"        "Report"       "RSS"          "RTS"
[51] "SASS"         "SBS"          "SCSORF"       "SDG"          "SRA"
[56] "SRBQP"        "SRSav"        "STAI"         "SWBQ"         "WEBEXEC"
```

If you need help creating a NEW task, see the section help creating a new task.

Below, a table with an overview of the available tasks:

Table 3.1: Test

short_name	Nombre	Descripcion
AIM	Grupos Socieeconómicos	Modelo que permite segmentar y clasificar socioeconómicamente a los individuos.
BART	Balloon Analogue Risk Task	Medida de riesgo individual mediante el análisis de las condultas del individuo
BNT	Berlin Numeracy test	Breve, flexible y de fácil aplicación. Mide habilidades numéricas
Bank	Detalles bancarios	Detalles bancarios para realizar transferencia a participante por su participación en experimento
CAS	COVID-19 anxiety scale	Evalúa las fluctuaciones en los niveles de ansiedad causados por COVID-19
CEL	Cuestionario Estilos de Liderazgo	Mide los estilos de liderazgo según tres criterios: Directivo, Democratico y Derivativo
CIT	Test de inferencia contrafactual	Prueba de selección múltiple que evalúa inferencia contrafactual.
COVIDCONTROL	Covid related Questions	Busca conocer los cambios que han ocurrido en la vida del entrevistado desde la aparición del COVID-19.

Table 3.1: Test *(continued)*

short_name	Nombre	Descripcion
CRS	The Centrality of Religiosity Scale	Mide la intensidad general de cinco dimensiones básicas de la religiosidad.
CRT7	Cognitive reflection test	Mide la tendencia a anular una alternativa de respuesta prepotente que es incorrecta y participar en una reflexión adicional que conduce a la respuesta correcta.
CRTMCQ4	Cognitive reflection test - 4 alternatives	Mide la tendencia a anular una alternativa de respuesta prepotente que es incorrecta y participar en una reflexión adicional que conduce a la respuesta correcta.
CRTv	Verbal Cognitive Reflection Test	Mide la capacidad de suprimir una intuición inicial (incorrecta) y reflexionar cognitivamente.
CS	Escala de Compasión	Mide la Compasión hacia otros
Consent	Consentimiento Informado de Participación	Consentimiento informado estandard
ConsentHTML	Consentimiento Informado de Participación	Consentimiento informado estandard
Cov19Q	Cuestionario Covid 19	Evalúa dimensiones relacionadas al covid-19
DASS21	Escalas de depresión ansiedad y estrés	Mide escala de ansiedad, estrés y depresión (autoinforme)
DEBRIEF	Debrief at end of experiment	Informe presentado al final del experimento
DEMOGR	Demographic scale	Ejemplo de tarea donde se recoje información demográfica del encuestado
EAR	Escala de Autoestima de Rosenberg	Instrumento unidimensional que mide la autoestima.
ERQ	Emotion Regulation Questionnaire	Evalua dos estrategias de regulación emocional (autoinforme)
ESM	Escala Subjetiva de Memoria	Evalúa la memoria del paciente directamente con él/ella
ESV	Empathy Stimulus Validation	Evalúa la interacción entre 2 individuos (uno con Alzeimer)
ESZ	Escala abreviada de Sobrecarga del Cuidador de Zarit	Mide la sobrecarga de los cuidadores de personas con Demencia (Autoinforme)

Table 3.1: Test *(continued)*

short_name	Nombre	Descripcion
EmpaTom	Empatia, Teoria de la mente y Compasión	Tarea de video social que permite la manipulación y evaluación independiente de la empatía y compasión
FONDECYT2022E1	Tarea experimental FONDECYT 2021 Gorka Navarrete	Tarea experimental para evaluar mejoras en la comprensión de PPV y NPV a partir de presentación de apoyo pictórico
GBS	Global Beliefs Screening	Evalúa las creencias del entrevistado
GHQ12	12-item General health questionnaire	Detecta personas con trastorno psiquiátrico diagnosticable
Goodbye	Goodbye task	Despedida del experimento. Entrega el código aleatorio de participación
HRPVB	High risk perception of vaginal birth	Preguntas para evaluar el nivel de riesgo percibido antes de ver la información de la prueba INFCONS
HRPVBpost	High risk perception of vaginal birth (POST)	Preguntas para evaluar el nivel de riesgo percibido después de ver la información de la prueba INFCONS
IBT	Impulsive buying Tendency	Mide impulsividad de compra
ICvsID	Impartial Charity vs. Instrumental Damage	Evalua las actitudes de las personas hacia el daño instrumental, es decir, el sacrificio de un individuo para salvar a un número mayor.
IDQ	Cuestionario de Identificación	Cuestionario que permite identificar a los individuos y además contiene preguntas sobre diagnósticos psiquiatricos.
IEC	Rotters internal-External control scale	Mide el grado en que las personas creen que ejercen control sobre sus vidas (controladas internamente) o el grado a los que sienten que sus destinos están más allá de su propio control y están determinados por el destino (controladas externamente)
INFCONS	Cesarean information brochures	Entrega información sobre las cesáreas (consentimiento informado)
IRI	Interpersonal reactivity index	Es una medida de empatía de diferencia individual con un enfoque multidimensional. (Medida de autoinforme)

Table 3.1: Test *(continued)*

short_name	Nombre	Descripcion
IRS	Importance of Rationality Scale	Mide la importancia personal (no verbal) que los individuos asignan a la formación y evaluación de la creencia sobre bases racionales
MDDF	MORAL DISGUST DUMBFOUNDING	Evalua el juicio moral, la toma de perspectiva y la valoración de daño percibido en cuatro tipos de casos:de un dilema con fuerte controversia moral (moral dumbfounding), dilemas morales sin fuerte controversia moral, situaciones no morales y situaciones control.
MDMQ	Melbourne Decision Making Questionnaire	Mide patrones de afrontamiento de decisiones
MIS	Magical ideation scale	Mide el nivel de pensamiento mágico del individuo
OBJNUM	Lipkus numeracy Test	Mide el rendimiento en preguntas de resolución numérica
OTRASRELIG	Varias preguntas sobre creencias religiosas	Escala con multiples preguntas sobre religión. Creada a partir de preguntas propias y multiples escalas.
PERMA	Perma Profiler	Medida de prosperidad o bienestar
PRFBM	Preferencia of birth mode in normal pregnancy	NA
PRFBMpost	Preferencia of birth mode in normal pregnancy (post)	NA
PSC	Prosociality	Variación del juego de la donación caritativa donde la persona decide, en el caso de una ganancia ocasional, cuanto dinero se queda y cuanto reparte a una entidad social que apoya a personas de su propio país (in-group) y cuanto a una entidad social que apoya a personas de otras partes del mundo (out-group).
PSETPP	Percepciones sobre el embarazo, trabajo de parto y parto (Fear of birth scale)	Evalúa las actitudes y sentimientos del individuo hacia el embarazo, trabajo de parto y parto.

Table 3.1: Test *(continued)*

short_name	Nombre	Descripcion
PSPPC	Percepciones sobre el parto por cesárea	NA
PSS	Perceived Stress Scale	Mide el grado en que las situaciones en la vida de uno son valoradas como estresantes.
PVC	Preguntas vacuna COVID-19	Entrega información respecto al estado de vacunación contra covid19
PWb	Psychological well being-scale	Mide el bienestar psicológico
R-PBS	Paranormal Belief Scale	Proporciona una medida del grado de creencia en siete dimensiones
REI40	Rational-Experiential intentory-40	Mide las diferencias individuales de dos dimensiones independientes del procesamiento humano: racional y experiencial.
RTS	The Revised Transliminality Scale	Mide el nivel de transliminalidad del entrevistado.
Report	Tarea para solicitar autorización y detalles para enviar reporte automatizado con resultados de participantes	La tarea Report solicita autorización y detalles para enviar reporte automatizado. Existe un archivo Rmd en jsPsychHelpeR que trabaja con el input de esta tarea para generar reportes automatizados para los participantes.
SASS	Social Adaptation Self-evaluation Scale	Estudia la motivación y el comportamiento del paciente en distintas áreas
SBS	Supernatural Belief Scale	Evalua la medida de la creencia en agentes religiosos sobrenaturales, lugares y eventos.
SCSORF	Santa Clara Strength of Religious Faith Questionnaire	Evalúa la fuerza de la fe religiosa.
SDG	Socio Demográfico General	Recolecta información demográfica del individio
SILS	School Implementation Leadership Scale	Evalúa estilos de liderazgo en el contexto educacional (directivos y profesores de institiuciones educacionales).
SRA	Self-Reported Altruism Scale	Mide el altruismo del individuo

Table 3.1: Test *(continued)*

short_name	Nombre	Descripcion
SRBQP	Self-regulation and belief questionnaire in pandemic	NA
SRSav	Springfield Religiosity Scale (Abbreviated Version)	Refleja una perspectiva judeocristiana conservadora o tradicional
STAI	State-Trait Anxiety Inventory	Entrega una medida empírica del nivel de ansiedad "normal" en adultos, medinate la medición de 2 conceptos independientes.
SWBQ	Spiritual wellbeing questionnaire	Mide el bienestar personal, bienestar comunitario, bienestar ambiental y bienestar trascendental.
WEBEXEC	Web based executive funtion questionnaire	Refleja la experiencia general de los participantes sobre los problemas ejecutivos en lugar de cualquier aspecto específico de los mismos.
bRCOPE	Escala de afrontamiento religioso Brief-RCOPE	Evalúa dos patrones de afrontamiento religioso en población adolescente y adulta
sProQOL	Short Professional Quality of Life Scale	Mide la Calidad de vida de trabajadores de la salud

3.2 Experiment configuration

In the `config.js` file you can find the main parameters to control how your experiment works.

You can edit the config file in one of the following two ways:

A) Go to `folder_output` and edit `config.js`.

B) Use the jsPsychMaker_config Shiny APP and copy the generated config.js file in the `folder_output` folder.

If you use the app, you will need to copy the generated config.js file to your protocol folder. The Shiny app can also help you create a parametrized consent form (see the Consent tab).

3.2.1 Main parameters

- `pid = 999;`: Number of protocol
- `online = true;`: true if the protocol runs in a server, false if it runs locally
- `max_participants = 3;`: If you have `between participants` conditions (participants are assigned to only one of a number of conditions), this is the max number of participants per condition
- `random_id = false;`: true if you want to assign a random id to participants, false if the participant needs to input an id
- `max_time = "24:00:00";`: Maximum time to complete the protocol (HH:MM:SS; Hours:Minutes:Seconds)
- `accept_discarded = true;`: If a participant is discarded (i.e. exceeded the max_time), should we allow them to continue, given there are available slots?
- `debug_mode = false;`: When testing the protocol:
- shows DEBUG messages
- creates the DB tables if they don't exist
- Avoids randomization (e.g. order of items) so the `jsPsychMonkeys` can have a reproducible behavior

3.2.2 Order of tasks

```
first_tasks = ['Consent'];//  The protocol will start with these tasks in sequential order
last_tasks = ['Goodbye'];//  Last block of tasks presented (in sequential order)
```

Create as many blocks as needed:

```
randomly_ordered_tasks_1 = ['TASK1', 'TASK2']; //  Block of tasks in random order
randomly_ordered_tasks_2 = ['TASK3']; //  Block of tasks in random order
secuentially_ordered_tasks_1 = ['TASK5', 'TASK4']; // Block of tasks in sequential order
```

The final array of tasks can be build combining the above blocks. The order of the tasks in the arrays starting with "random" will be randomized.

```
tasks = ['first_tasks',
         'randomly_ordered_tasks_1',
         'secuentially_ordered_tasks_1',
```

```
        'randomly_ordered_tasks_2',
        'last_tasks'];
```

3.2.3 Between-subject tasks

The variable `all_conditions` in `config.js` let's you define the Independent Variables (IV) and levels for the between-subject tasks:

If there is no between-subject task:

```
all_conditions = {"protocol": {"type": ["survey"]}};
```

If there are between-subject tasks:

```
all_conditions = {"NAMETASK": {"name_IV": ["name_level1", "name_level2"]}};
```

`jsPsychR` will randomize participants to the different conditions keeping the unbalance between conditions to the minimum possible.

3.3 online-offline protocols

jsPsych uses standard web technologies (HTML, CSS y Javascript), so that protocols should run in any modern browser (updated, please). We recommend Google Chrome just because our test suite runs with Google Chrome, so we will catch its specific issues earlier.

3.3.1 Offline

If you want to run a protocol locally (on your computer, on a lab computer), you need to:

- set `online = false;` in the `config.js` file
- double click index.html

`jsPsychR` will use `IndexedDB` to store the participants' progress and balance between conditions. The output csv files will be Downloaded to the Download folder of the computer where the protocol runs.

3.3.1.1 CORS ERRORS

If any of the tasks imports an html file, the Offline protocol will give a CORS error.

There are ways to disable web security in your browser, but it **MUST only be done if your experiment computer runs offline**, otherwise you will be exposed to very bad things.

See how to run chrome disabling web security to avoid CORS error:

- `google-chrome --disable-web-security --user-data-dir="~/"`

3.3.2 Online

Tu run a protocol online, set `online = true;` in the `config.js` file. You will need a couple more things:

- MySQL running in your server
- A file `.secrets_mysql.php` with the content below
- Define the route to `.secrets_mysql.php` in `controllers/php/mysql.php`
  - `require_once '../../.secrets_mysql.php';`
  - THIS FILE **MUST NOT** BE PUBLICLY VISIBLE FROM THE BROWSER
- **Upload the files to the server** :)

```php
<?php

/* DO NOT UPLOAD TO PUBLIC REPO */

  $servername = "127.0.0.1";
  $username = "USERNAME OF THE DATABASE";
  $password = "PASSWORD OF THE DATABASE";
  $dbname = "NAME OF THE DB";

?>
```

`jsPsychR` will use `MySQL` to store the participants' progress and balance between conditions. The output csv files will be Downloaded in the `.data/` folder inside the protocol folder in the server.

Before launching the final experiment, make sure you start with a clean slate! That can be summarized in 3 simple steps:

1. Check the configuration for you experiment (`config.js`) and make sure all is well. Some of the critical bits are:

```
pid = 999; // SHOULD have your project ID!
online = true; // true is good
max_participants = 100; // Max participants per contition [number]
max_time = "24:00:00"; // Max time to complete the protocol [HH:MM:SS]
debug_mode = false; // SHOULD be false
```

2. Check that the `.data/` folder for your protocol is empty in the server. You will likely have remains of the piloting and Monkeys.
3. Clean up the MySQL data associated to your protocol.

```
SET @PID = 999; // HERE YOUR PROTOCOL ID!

delete from experimental_condition where id_protocol=@PID;
delete from user where id_protocol=@PID;
delete from user_condition where id_protocol=@PID;
delete from user_task where id_protocol=@PID;
delete from task where id_protocol=@PID;
delete from protocol where id_protocol=@PID;
```

You will most likely need help from the server admin to perform these steps.

3.4 Need help implementing a task!

If you need help creating a NEW task, see the section help creating a new task.

3.5 Developing tasks

Remember to place an `if (debug_mode === false)` before the randomization of the item order so when running in debug_mode, the items are not randomized. This is important so the behaviour of the jsPsychMonkeys is reproducible:

```
if (debug_mode === false) NAMETASK = jsPsych.randomization.repeat(NAMETASK,1);
```

3.6 Technical aspects

We currently use jsPsych 6.3, and plan to migrate to the last stable jsPsych at some point. There is a migration guide and a Github issue with migration questions.

3.6.1 Misc

When index.html is launched:

- Checks if there are available slots

When an uid is assigned:

- `questions` array is created
- `between-participants` conditions are assigned and stored in the DB (MySQL if online, IndexedDB if offline)

Each question, timeline or conditional question needs to have a:

```
data: {trialid: 'NameTask_001', procedure: 'NameTask'}
```

The `trialid` identifies the trial, and the `procedure` makes possible to find that trial so participants can continue the tasks where they left, know when participants finished the tasks, etc. This is done in MySQL if online, IndexedDB if offline.

`trialid`'s need to have a standardized structure, which generally conforms with `NameTask_3DigitNumber`. When using conditional items the structure can be a bit more complex, but not much. We use the following rules to check for non-complying trialid's:

```
^[a-zA-Z0-9]{1,100}_[0-9]{2,3}$ -> `NameTask_2or3DigitNumber`, for example `BNT_001`
^[a-zA-Z0-9]{1,100}_[0-9]{2,3}_[0-9]{1,3}$ -> `NameTask_2or3DigitNumber_1to3DigitsSuffix`, fo
^[a-zA-Z0-9]{1,100}_[0-9]{2,3}_if$ -> `NameTask_2or3DigitNumber`, for example `BNT_002_if`
^[a-zA-Z0-9]{1,100}_[0-9]{2,3}_[0-9]{1,3}_if$  -> `NameTask_2or3DigitNumber`, for example `BN
```

3.6.2 jsPsychMaker main changes on a task

1. Start of a task

```
questions = ( typeof questions != 'undefined' && questions instanceof Array ) ? questions
questions.push( check_fullscreen('NameOfTask') );
NameOfTask = [];
```

2. Each item

```
data: {trialid: 'NameOfTask_01', procedure: 'NameOfTask'}
```

3. End of experiment

```
if (debug_mode == 'false') NameOfTask = jsPsych.randomization.repeat(NameOfTask, 1);
NameOfTask.unshift(instruction_screen_experiment);
questions.push.apply(questions, NameOfTask)

questions.push({
    type: 'call-function',
    data: {trialid: 'NameOfTask_000', procedure: 'NameOfTask'},
    func: function(){
      if (online) {
        var data = jsPsych.data.get().filter({procedure: 'NameOfTask'}).csv();
      } else {
        var data = jsPsych.data.get().filter({procedure: 'NameOfTask'}).json();
      }
      saveData(data, online, 'NameOfTask');
    }
});
```

3.6.3 Conditional questions

```
var question001 = {
  type: 'survey-multi-choice-vertical',
  questions: [{prompt: '<div class="justified">¿Usted se ha vacunado contra el coronavirus
  data: {trialid: 'PVC_001', procedure: 'PVC'}
};
PVC.push(question001);
```

```
var question001_1 = {
  type: 'survey-multi-choice-vertical',
  questions: [{prompt: '<div class="justified">¿Usted se va a vacunar contra el coronaviru
  data: {trialid: 'PVC_001_1', procedure: 'PVC'}
};

var if_question001_1 = {
  timeline: [question001_1],
  data: {trialid: 'PVC_001_1_if', procedure: 'PVC'},
  conditional_function: function(){
    let data = (JSON.parse((jsPsych.data.get().values().find(x => x.trialid === 'PVC_001')
    if((data) ==  'No'){
      return true;
    } else {
      return false;
    }
  }
};
PVC.push(if_question001_1);
```

3.7 Common ERRORS

If you get the following error in the console: `Uncaught TypeError: Cannot read properties of undefined (reading 'procedure')`

Run this in the console:

```
for (var i = 0; i < questions.length; i++) {
  console.log(i + questions[i].data["procedure"])
}
```

It will stop in one of the items. Go to the console, check the array `questions` and go to the number that failed.

When you know the task and item that fails, you probably need to add:

```
`data: {trialid: 'TASKNAME_ITEMNUMBER', procedure: 'TASKNAME'}
```

4 jsPsychMonkeys

---

jsPsychMonkeys: Release Monkeys to a jsPsych experiment using the R package {targets}, docker and {RSelenium}.

In brief

With jsPsychMonkeys you can:

- Simulate participants online and offline
- Simulate participants sequentially and in parallel
- Ask your Monkeys to take pictures of each screen of the protocol
- Make the behavior of the Monkeys reproducible setting a random seed associated with their unique id
- Store logs of the process, including console logs with errors
- Watch your participants randomly click things in VNC (you will need to install realvnc)

See QuickGuide for basic instructions.

Setup

Ubuntu

You may need to install some system libraries first:

- `sudo apt install libssl-dev libcurl4-openssl-dev libxml2-dev docker`
- If the Monkeys do their work but no csv's appear, make sure your the docker user has write access to the `~/Downloads` folder.

Windows

- Install docker desktop
- Update wsl (in a command prompt): `wsl - update`

Mac

- Install docker desktop

4.1 How to simulate participants

To run a monkey locally:

```
jsPsychMonkeys::release_the_monkeys(uid = 1,
                                    local_folder_tasks = "~/Downloads/protocol999/")
```

To run a monkey on a server:

```
jsPsychMonkeys::release_the_monkeys(uid = 1,
                                    server_folder_tasks = "999",
                                    credentials_folder = "~/.vault/")
```

`credentials_folder` must contain `SERVER_PATH.R` and `.credentials`. See below for the expected content of those files.

4.2 Parameters available

There are a few parameters for `jsPsychMonkeys::release_the_monkeys()` that can be useful:

- `uid_URL = TRUE`: The uid is passed in the URL (e.g. `&uid=1`)
- `local_folder_tasks = rep("Downloads/tests/test_prototol", 25)`: Passing a vector of multiple protocols will make the Monkeys to complete all of them.
- `times_repeat_protocol`: How many times a monkey should complete the same protocol (useful for longitudinal protocols or to speed up things)
- `time_to_sleep_before_repeating_protocol`: How many seconds to wait before reattempting to complete the protocol
- `keep_alive = TRUE` Keep the docker container alive after completing the tasks
- `DEBUG = TRUE` Activate DEBUG mode. Lot's of stuff will show up in the console.
- `open_VNC = TRUE` Activate DEBUG mode and open a VNC container to see the Monkeys' progress.
- `screenshot = TRUE` The Monkeys will take a picture of all the pages they see. The .png files are stored in `outputs/screenshots`
- `debug_file = TRUE` Activate DEBUG mode and store all the console output in the `outputs/log`
- `big_container = TRUE` Sets the Shared memory size (/dev/shm) to 2 gigabytes. This is useful to avoid long/complex protocols to crash
- `disable_web_security = TRUE` If you are running a local protocol that loads external files (e.g. consent form in a html file), you may need this. Only works with Google Chrome.
- `console_logs = TRUE` Store the browser's console logs. Only works with Google Chrome
- `forced_random_wait = TRUE` Will wait a randomly sampled number of seconds on page 4
- `forced_seed = 11` Set a random seed so the Monkeys' behavior will be fully reproducible
- `forced_refresh = 20` Refresh browser in page 20 (if TRUE is given, it will refresh in a randomly sampled page)
- `sequential_parallel` Choose between `sequential`, the default, or `parallel`
- `number_of_cores` Number of cores for parallel monkeys. The default is half of the available cores

4.2.1 Parameters details

- `local_folder_tasks`: If the folder is not accessible to Docker (anything outside the Download folder), jsPsychMonkeys will create a copy of the protocol in `Downloads/JSPSYCH/`

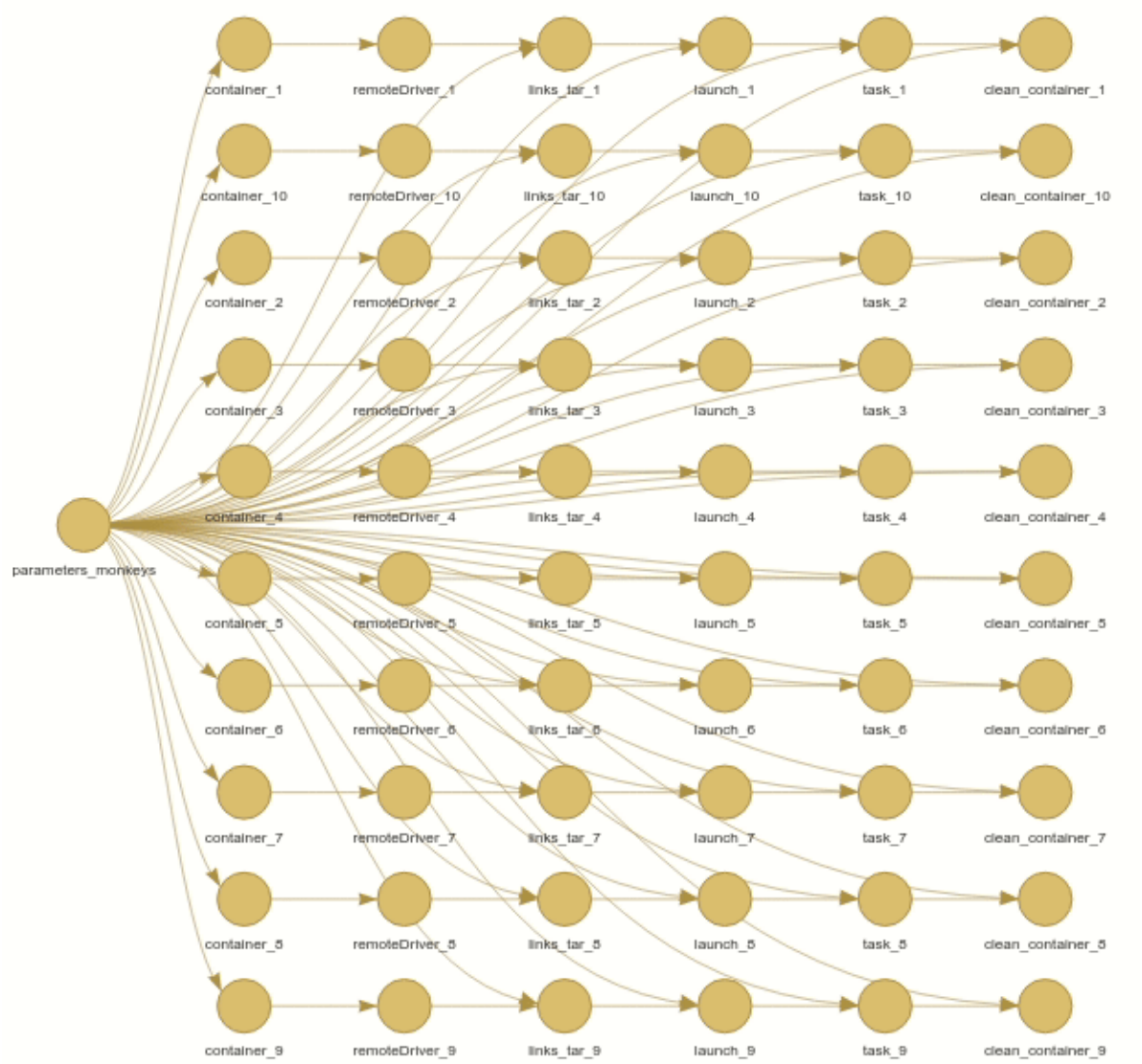
4.3 Release a horde of Monkeys!

If you want a horde of Monkeys, you can set up `sequential_parallel = "parallel"` and choose how many monkeys will run in parallel with `number_of_cores`:

```
jsPsychMonkeys::release_the_monkeys(uid = "1",
                                    local_folder_tasks = "~/Downloads/protocol999/",
                                    sequential_parallel = "parallel",
                                    number_of_cores = 4)
```

---

10 Monkeys completing a protocol in parallel:



4.4 Issues

If the setup configuration steps didn't work... You may need to do one of the things below:

- Switch to Ubuntu :-)
- Run participants manually

4.5 Technical aspects

4.5.1 Launch Monkeys on a server

You will need two files for the configuration in the hidden and NOT SHARED `.vault/` folder:

- `.vault/SERVER_PATH.R`: contains the path where the protocols are located in your server: `server_path = "http://URL_OF_YOUR_SERVER/PROTOCOLS_GENERAL_FOLDER/"`
- `.vault/.credentials`: contains a list with the user and password for the server: `list(user = "", password = "")`

With the `server_folder_tasks` you will set the subfolder where the protocol is located. In the example below the Monkeys would go to, `http://URL_OF_YOUR_SERVER/PROTOCOLS_GENERAL_FOLDER/999`

4.5.2 Alternatives

Since jsPsych 7.1 there is a simulation mode available, which should be much faster than the good ol' Monkeys. Once we migrate to jsPSych 7.x, we may retire this section.

5 jsPsychHelpeR

---

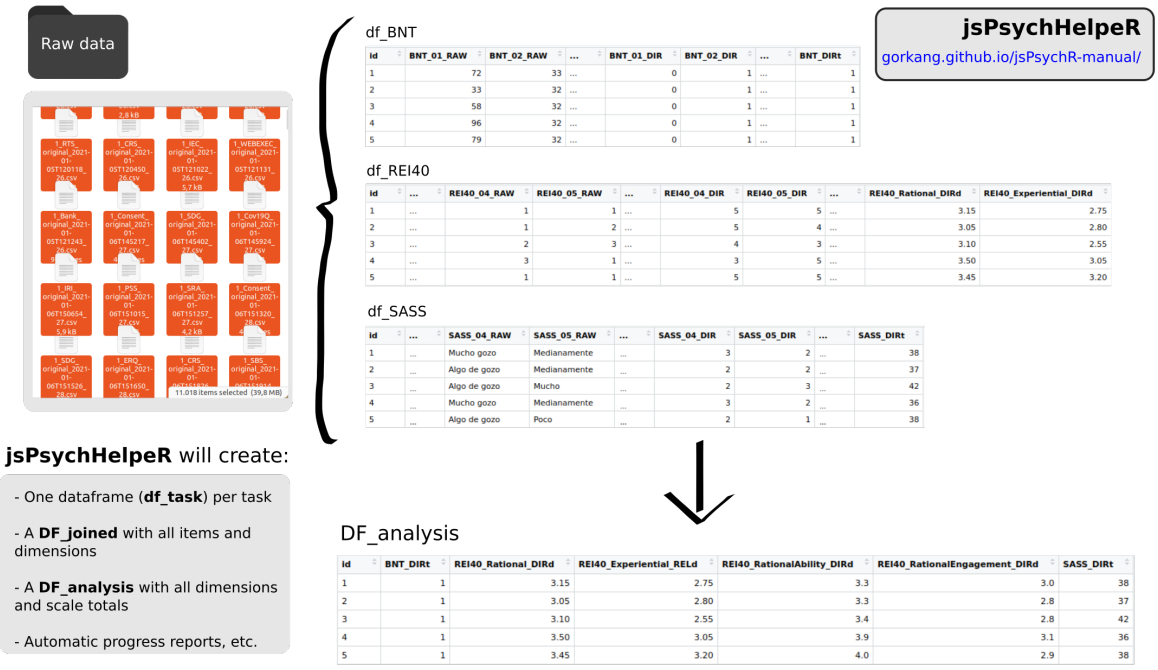
jsPsychHelpeR: Standardize and automatize data preparation and analysis of jsPsych experiments created with jsPsychMaker.

In brief

jsPsychHelpeR will lend you a hand automatizing and standardizing your data preparation and analysis.

- Use a completely open, reproducible and automatic process to prepare your data
- Data preparation ready for > 70 tasks (see here the available tasks)
- Get tidy output dataframes for each task, and for the whole protocol
- Include tests for common issues
- Automatic reports with progress, descriptives, codebook, etc.
- Create a fully reproducible Docker container with the project's data preparation and analysis
- Create a blinded dataframe to be able to perform blinded analyses. See 10.1038/526187a and 10.1177/25152459221128319

See QuickGuide for basic instructions.



5.1 How to prepare data

Our goal is that each jsPsychMaker task has a sister script on jsPsychHelpeR to help prepare the data automatically. If a task you need does not have one, you can try to create the script yourself and do a pull request in the jsPsychHelpeR repo, or see the section help creating a new task.

If you already ran a pilot experiment, simply:

1. Install `jsPsychHelpeR`:

```
if (!require('remotes')) utils::install.packages('remotes'); remotes::install_github('gork
```

2. `jsPsychHelpeR::run_initial_setup()` will:

- Try to make sure you have all the dependencies, folders, etc.
- Copy the data to the data/pid folder
- Move data with sensitive tasks to the .vault folder

- Create a customized `_targets.R` file adapted to the data of your protocol, so data preparation can run automagically

```
jsPsychHelpeR::run_initial_setup(pid = '999',
                                 data_location = '~/Downloads/JSPSYCH/999/',
                                 folder = '~/Downloads/jsPsychHelpeR_999/')
```

If you have the sFTP credentials for the server, it will:

- Download all the data from your protocol (you will need the FTP credentials and set `download_files = TRUE`)

- Download and zip a copy of the full protocol without the data (you will need the FTP credentials and set `download_task_script = TRUE`)

This should work on Ubuntu, if you have the FTP credentials, and `sshpass` and `rsync` installed.

```
jsPsychHelpeR::run_initial_setup(pid = '999',
                                 download_files = TRUE,
                                 download_task_script = TRUE,
                                 folder = "~/Downloads/jsPsychHelpeR_999")
```

5.1.1 Targets pipeline

To help make the pipeline reproducible and more efficient, we use the targets package (Landau 2021). A few basic things to know:

- **The whole process can be reproduced running `targets::tar_make()`**
- A nice visualization of all the pre-processing steps can be seen with `targets::tar_visnetwork(targets_o = TRUE)`
- The file `_targets.R` contains the important parameters and calls to all the functions used when running `targets::tar_make()`

To see more detail about any specific step, you can:

1. Go to the relevant function in `_targets.R` (cursor on a function, then F2)

2. Load the input parameters of the function with `debug_function(NAME_OF FUNCTION)`. Alternatively, manually use `targets::tar_load(NAME_OF_TARGET)`
3. Run the code step by step as you would normally do

5.2 Basics

jsPsychHelpeR uses as input a data created with a jsPsychMaker experimental protocol.

5.2.1 Inputs

The input data folder will be named after the protocol_id, for example `999/` and needs to be placed in the `data/` folder of the jsPsychHelpeR project `data/YOUR_PROJECT_NUMBER`:

- The data folder can contain either multiple .csv files, or a single .zip file

There will be a single .csv file for each participant and task of the protocol. For example:

- 999_Consent_original_2022-04-02T205622_1.csv:
  - [project: 999]_[experimento: Consent]_[version: original]_[datetime: 2022-04-02T205622]_[participant id: 1]

5.2.2 Outputs

When the pipeline successfully runs with `targets::tar_make()`, a number of outputs will be created.

All the outputs can be found in the `/outputs` folder. The only exception is the sensitive data and reports, which can be found in `.vault/outputs`. WARNING: The '.vault/' folder **MUST NOT** be made public.

5.2.2.1 Output folders

The outputs will be organized in different folders:

- **Data frames** for different stages of data processing can be found in `outputs/data`
- **Temporary files for manual correction** are in `outputs/data/manual_correction` (the final manual correction files must be place by the user in `data/manual_correction`). WARNING: These will be overwritten each time the pipeline runs

- **Plots**, **tables** and **reports** are in `outputs/plots`, `outputs/tables`and `outputs/reports` respectively.
- **Test outputs** are in `outputs/tests_outputs`
- **Anonymized Raw data** will be moved to `.vault/data_vault/`

5.2.2.2 Output dataframes

There will be a single data frame (df) for each of the tasks in `outputs/data`, plus a data frame (DF) for each of the steps of the data preparation, and a dictionary file listing all the available tasks. We store the files in two formats, csv and rds:

- **DF_raw.csv**: All the `data/project_id/` csv files combined on a single file. We only add the columns "project", "experimento", "version", "datetime", "id" by parsing the filenames
- **DF_clean.csv**: Clean version of the raw file ready to process the individual tasks
- **df_ShortNameOfTask.csv**: One df for each of the tasks of the protocol after being processed with the `prepare_ShortNameOfTask()` functions
- **DF_joined.csv**: all the processed tasks joined in a single DF
- **DF_analysis**: only the total scores and dimensions from `DF_joined` (columns ending in `_DIRt`, `_STDt`, `_DIRd`, `_RELd`, `STDd`). Can be visually explored using the shiny app in `Rmd/app.R`
- **DF_analysis_blinded**: If the `DVars` parameter of `create_DF_analysis()` is not empty, jsPsychHelpeR will create DF_analysis_blinded where the DVars will be scrambled so te data analysts can perform blinded analysis
- **DICCIONARY_tasks.csv**: list of all tasks in the protocol

5.2.2.3 Output dataframes column names

All the output processed data frames columns are named in a standardized way:

- **ShortNameOfTask_ItemNumber_RAW**: raw responses of participants for individual items
- **ShortNameOfTask_ItemNumber_DIR**: processed raw responses following the task correction instructions (e.g. inverting certain items, converting strings to numbers, computing accuracy…)
- **ShortNameOfTask_RAW_NA**: number of missing data (NA) in the RAW responses

- **ShortNameOfTask_DIR_NA**: number of missing data (NA) in the DIR responses. If it is not equal to `ShortNameOfTask_RAW_NA` there is something wrong in the items correction.
- **ShortNameOfTask_DimensionName_DIRd**: scores for a specific dimension (`d`) in a task, calculated following task correction instructions (e.g. summing or averaging certain items)
- **ShortNameOfTask_DimensionName_RELd**: scores for a specific dimension (`d`) in a task, calculated following task correction instructions AND after filtering items with low reliability. See Reliability section for more information.
- **ShortNameOfTask_DimensionName_STDd**: standardized score for a dimension (`d`)
- **ShortNameOfTask_DIRt**: total (`t`) score for a task calculated following task correction instructions (e.g. summing or averaging all items)
- **ShortNameOfTask_STDt**: standardized (`t`) score for a task

5.3 Errors in the pipeline

See the targets manual for more information.

We include `tar_option_set(workspace_on_error = TRUE)` in `_targets_options.R` so if there is an error in our pipeline, `targets` will automatically save the workspace. This allows you to go to the relevant target and debug interactivelly.

If you get an error:

1. List the available workspaces (e.g. DF_clean):

```
tar_workspaces()
```

2. Load the errored workspace:

```
tar_workspace(DF_clean)
```

5.4 Advanced

5.4.1 Need help preparing new task

If you need help preparing a NEW task, see the section help with new tasks.

5.4.2 Create your own reports

You can use any of the template reports in the `_targets.R` file, or create your own reports.

We will start opening one of the template reports: `rstudioapi::navigateToFile("doc/report_analysis.Rmd'`

- Edit the RMarkdown file to adapt it to your needs.
- If you already did `targets::tar_make()`, when running `targets::tar_load(DF_analysis)` the dataframe `DF_analysis` will load in your Environment.

Go back to the `_targets.R` file:

- Look for `# Analysis report` and uncomment the following lines:

```
# tar_render(report_analysis, "doc/report_analysis.Rmd",
#            output_file = paste0("../outputs/reports/report_analysis.html")),
```

When you finished editing and uncomented the tar_render command, go back to the `run.R` file:

- `targets::tar_make()`

5.4.3 Create new tasks

To create the correction script for a new task, you start with:

- `create_new_task(short_name_task = "NAMETASK")`

This will:

- create a new file from a template correction script (`R_tasks/prepare_TEMPLATE.R`)
- adapt it to your `short_name_task` to make everything as standardized as possible
- open the new `prepare_NAMETASK.R` file

If the parameter `get_info_googledoc = TRUE`:

- The NEW tasks document is checked.
- If the document has been filled properly, it will show in the console standardized strings (ready to be copy/pasted to the new `prepare_NAMETASK.R` file) about:
  - dimension names
  - items corresponding to each dimension

- dimension calculation
- inverse items
- numeric conversion of items

You can also use `get_dimensions_googledoc()` as a standalone function:

```
get_dimensions_googledoc(short_name_text = "MLQ")
```

All the `prepare_NAMEOFTASK.R` scripts on the `R_tasks/` folder have been created starting from the same template. The only exception are the experimental tasks and some surveys with particularities that require more complex adaptations.

When you finish implementing the correction script, please do a Pull request so we can add you script to the pool. If you have not already, please help us filling up details about the task in the NEW tasks document.

5.4.4 Adapting new tasks

`get_dimensions_googledoc` will show you how to adapt the `prepare_TASK()` script, but you will need to know how it works to be able to edit the relevant bits. Also, sometimes `get_dimensions_googledoc` won't get all the details of the task right, or there could be non-standard elements to it. Here, we will describe some of the elements of the template to help understand how it works.

Remember you should **ALWAYS** start with `create_new_task(short_name_task = "NAMETASK")` so your task template works well with `jsPsychHelpeR`.

There are three chunks you will need to adapt to have a fully working preparation script.

- `[ADAPT 1/3]: Items to ignore and reverse, dimensions`
- `[ADAPT 2/3]: RAW to DIR for individual items`
- `[ADAPT 3/3]: Scales and dimensions calculations`

5.4.4.1 Items to ignore and reverse, dimensions

```
# [ADAPT 1/3]: Items to ignore and reverse, dimensions ------------------------
# ****************************************************************************

description_task = "" # Brief description here

items_to_ignore = c("000") # Ignore these items: If nothing to ignore, keep as is
items_to_reverse = c("000") # Reverse these items: If nothing to reverse, keep as is
```

```
## NameDimension1, NameDimension2 should be the names of the dimensions
## Inside each c() create a vector of the item numbers for the dimension
## Add lines as needed. If there are no dimensions, keep as is
items_dimensions = list(
  NameDimension1 = c("000"),
  NameDimension2 = c("000")
)

# [END ADAPT 1/3]: ************************************************************
# *****************************************************************************
```

5.4.4.2 RAW to DIR for individual items

```
DF_long_DIR =
    DF_long_RAW %>%
    select(id, trialid, RAW) %>%



  # [ADAPT 2/3]: RAW to DIR for individual items --------------------------------
  # *****************************************************************************

    # Transformations
    mutate(
      DIR =
        case_when(
          RAW == "Nunca" ~ 1,
          RAW == "Poco" ~ 2,
          RAW == "Medianamente" ~ 3,
          RAW == "Bastante" ~ 4,
          RAW == "Mucho" ~ 5,
          is.na(RAW) ~ NA_real_,
          grepl(items_to_ignore, trialid) ~ NA_real_,
          TRUE ~ 9999
        )
    ) %>%

    # Invert items
    mutate(
      DIR =
```

```
        case_when(
          DIR == 9999 ~ DIR, # To keep the missing values unchanged
          trialid %in% paste0(short_name_scale_str, "_", items_to_reverse) ~ (6 - DIR),
          TRUE ~ DIR
        )
    )

  # [END ADAPT 2/3]: ************************************************************
  # *****************************************************************************
```

5.4.4.3 Scales and dimensions calculations

```
# [ADAPT 3/3]: Scales and dimensions calculations ---------------------------
# *****************************************************************************

# Reliability ---------------------------------------------------------------
# REL1 = auto_reliability(DF_wide_RAW, short_name_scale = short_name_scale_str, items = it
# items_RELd1 = REL1$item_selection_string


# [USE STANDARD NAMES FOR Scales and dimensions: names_list$name_DIRd[1], names_list$name_
# CHECK with: create_formulas(type = "dimensions_DIR", functions = "sum", names_dimensions
DF_wide_RAW_DIR =
  DF_wide_RAW %>%
  mutate(

    # [CHECK] Using correct formula? rowMeans() / rowSums()

    # Score Dimensions (see standardized_names(help_names = TRUE) for instructions)
    !!names_list$name_DIRd[1] := rowMeans(select(., paste0(short_name_scale_str, "_", item
    !!names_list$name_DIRd[2] := rowSums(select(., paste0(short_name_scale_str, "_", items

    # Reliability Dimensions (see standardized_names(help_names = TRUE) for instructions)
    # !!names_list$name_RELd[1] := rowMeans(select(., paste0(short_name_scale_str, "_", it

    # Score Scale
    !!names_list$name_DIRt := rowSums(select(., matches("_DIR$")), na.rm = TRUE)

  )
```

```
# [END ADAPT 3/3]: ************************************************************
# *****************************************************************************
```

5.4.5 DEBUG tasks

At the begining of each of the `R_tasks/prepare_NAMETASK.R` scripts you will find a commented `debug_function(prepare_NAMETASK)` line.

When running it, it will load the input parameters for the task. From there, you can work inside of the preparation scipt as you would normally do in a R script.

If you get the error `"Error in debug_function(prepare_NAMETASK) : could not find function 'debug_function'`debug_function()`does nor work"` you will need to load all the functions in the `R/` folder first.

You can do this in one of three ways:

- `CONTROL + P` shortcut will work if the `run_initial_setup()` completed correctly (at least on Ubuntu systems).
- Run `targets::tar_load_globals()`
- Or directly, source all the scripts in the `R/` folder: `invisible(lapply(list.files("./R", full.names = TRUE, pattern = ".R$"), source))`

5.4.6 Docker containers

The function `jsPsychHelpeR::create_docker_container()` will create a fully reproducible docker container with the data preparation and analysis for a specific project.

The container can be easily shared or stored to allow others to run the data preparation and analysis for you project without worrying about dependencies, versions of packages, etc.

See more information about the setup in the admin section.

The gist of it is, after you have the full data preparation and analysis for you project ready, to create the container image and share it, just run:

```
# 1) Set your project ID
  PID = 999

# 2) Create docker image
  jsPsychHelpeR::create_docker_container(PID = PID)
```

```
# 3) SHARE your docker image

  # Using Dockerhub
    system(paste0("docker push gorkang/jspsychhelper:pid", PID))

  # Using a .tar file
    system(paste0("docker save gorkang/jspsychhelper:pid", PID, " | zip > pid", PID, ".tar
```

To load and run the container image (if you are using Windows, see here):

```
# 1) Set your project ID
  PID = 999

# 2) Get the docker image loaded into to your computer

  # Dockerhub
  system(paste0("docker pull gorkang/jspsychhelper:pid", PID))

  # .tar file
  utils::unzip(zipfile = paste0("pid", PID, ".tar.zip"), files = paste0("-"))
  system(paste0("docker load --input -"))

# 3) Run docker container
  system(paste0("docker run --rm -d --name pid", PID, " -v ~/Downloads/jsPsychHelpeR", PID
```

The output will be in `Downloads/jsPsychHelpeR[PID]/outputs/` after a couple of minutes. You can see the data preparation and analysis progress using docker desktop.

5.4.7 Blinded analysis

The function `create_DF_analysis()` has the parameter `DVars` to select the Dependent Variables in your data that should be scrambled to be ready for a blinded analysis. We use a simple `sort()` in those variables, so their data will be ordered from smaller to bigger, losing the relationship with the other variables in the data, but keeping their structure.

See MacCoun, R., & Perlmutter, S. (2015). Blind analysis: Hide results to seek the truth. Nature, 526(7572), 187-189 (https://doi.org/10.1038/526187a), or Sarafoglou, A., Hoogeveen, S., & Wagenmakers, E. J. (2023). Comparing analysis blinding with preregistration in the many-analysts religion project. Advances in Methods and Practices in Psychological Science, 6(1), 25152459221128319. (https://doi.org/10.1177/25152459221128319)

5.5 Helper functions

5.5.1 Reliability

You can use the `auto_reliability()` function to help you automatically filter items with low reliability (although doing this automatically is probably a bad idea). The function uses `psych::alpha()` and filters by default items with an r.drop <= 0.2. See `psych::alpha()` help for more details. **IMPORTANT**: Using `psych::omega()` is generally a better idea, see the alpha help page.

An example can be found in prepare_REI40().

The basic logic would be:

```
# Define items for a specific dimension
items_DIRd1 = c("01", "02", "03", "04", "05", "06", "07", "08", "09", "10")

# Calculate reliability
REL1 = auto_reliability(DF_wide_RAW, short_name_scale = short_name_scale_str, items = item

# Store item selection in a variable
items_RELd1 = REL1$item_selection_string

# In the final Dimension calculation, use the item selection including only the items with
## See `items_RELd1` below
!!names_list$name_RELd[1] := rowMeans(select(., paste0(short_name_scale_str, "_", items_RE

# Compare it with the calculation including the original items
## See `items_DIRd1` below
!!names_list$name_DIRd[1] := rowMeans(select(., paste0(short_name_scale_str, "_", items_DI
```

5.6 Technical aspects

5.6.1 How trialid's are processed

See `PRFBM`:

- If more than one response per screen
  - Item: `PRFBM_04`
  - Responses: `{"daño":"Parcialmente en desacuerdo","beneficio":"Parcialmente en desacuerdo"}`

- final trialids: `PRFBM_04_beneficio` and `PRFBM_04_daño`

5.7 Common ERRORS

5.7.1 `run_initial_setup()`:

```
x Can find server credentials in '.vault/.credentials'
x 0 tasks found for protocol 'TU NUMERO DE PROYECTO'. NOT creating _targets.R file
```

5.7.1.1 On Linux (Ubuntu):

- IF you have the server credentials:
  - Open .credentials_TEMPLATE `rstudioapi::navigateToFile(".vault/.credentials_TEMPLATE"`
  - Edit the file with your server credentials
  - Rename the file to `.credentials`

- IF you DON'T have the credentials but you have the .csv results files:
  - Copy the csv files to the folder `data/YOUR_PROJECT_NUMBER`
  - Run again `run_initial_setup()`

5.7.1.2 On Mac or Windows:

- Copy the csv files to the folder `data/YOUR_PROJECT_NUMBER`
- Run again `run_initial_setup()`

5.7.2 Rendering Rmd's

> Error: ! missing files _targets/meta/meta Execution halted

It is better to run everything, including your reports, inside the pipeline (`targets::tar_make()`).

If you need to the knitr (or render) button, you will have to:

- Load DF's `DF_analysis = readr::read_rds(here::here("_targets/objects/DF_analysis"))` instead of `targets::tar_load(DF_analysis)`
- Include all the necessary `library()` calls

> Error : Could not parse knitr report Rmd/report_analysis.Rmd to detect dependencies: Scanner error: mapping values are not allowed in this context at line 6, column 17

There is something wrong in Your YAML heather.

6 jsPsychR Admins

Instructions and protocols for jsPsychR admins. The goal is to minimize issues and make sure tasks behave in a consistent and reproducible manner.

6.1 jsPsychAdmin

---

jsPsychAdmin: Functions to help with common administrative tasks for jsPsychR protocols

---

- Install with:

```
# if (!require('renv')) utils::install.packages('renv'); renv::install("gorkang/jsPsychAdm
```

You will need a `.vault` folder with `.credentials`, `data_encrypted.rds` and `data_public_key.txt`.

6.1.1 Common tasks

6.1.1.1 CHECK participants all protocols

```
# Uses the .credentials file + the public key to unlock the encrypted data_encrypted.rds
jsPsychAdmin::check_status_participants_protocol()
```

6.1.1.2 Clean up a DEV protocol

```
# Clean up DB and csv files for a test/protocols_DEV/ protocol # Useful when testing
# rstudioapi::navigateToFile(".vault/.credentials")
jsPsychAdmin::clean_up_dev_protocol(protocol_id = "test/protocols_DEV/31") # Asks for se
```

6.1.1.3 Check missing scripts

```
#  Downloads all the protocols to CSCN-server
#  Then checks:
# - for which ones we do not have preparation scripts
# - for which ones we do not have googledoc details in
# + https://docs.google.com/spreadsheets/d/1Eo0F4GcmqWZ1cghTpQlA4aHsc8kTABss-HAeimE2IqA/
# + https://docs.google.com/spreadsheets/d/1LAsyTZ2ZRP_xLiUBkqmawwnKWgy8OCwq4mmWrrc_rpQ/

jsPsychAdmin::check_missing_prepare_TASK(sync_protocols = TRUE, check_trialids = TRUE, c
                                         gmail_account = "myemail@gmail.com")

# Will sync all protocols on server to the LOCAL folder ../CSCN-server
# Then, will CHECK:
# - Tasks with no prepare_TASK() script!
# - Tasks NOT in Google Doc
# - Check trialid's are OK
# - Check no missing info in Google doc of NEW tasks
jsPsychAdmin::download_check_all_protocols(gmail_account = "myemail@gmail.com")
```

6.1.1.4 Build jsPsychHelpeR and jsPsychMonkeys packages

Scripts to build the template projects for jsPsychHelpeR and jsPsychMonkeys

```
jsPsychAdmin::create_jsPsychHelpeR_zip()
jsPsychAdmin::create_jsPsychMonkeys_zip()
```

6.2 Docker containers

We can create a fully reproducible docker image with the data preparation and analysis for a specific project using `jsPsychHelpeR::create_docker_container()`

Afterwards, you can use the image to run a docker container that will reproduce the analysis and results of your project.

The current version is a first attempt at this, so there is a lot to improve.

6.2.1 Install Docker

First, we need to install docker.

- Linux: follow installation instructions
- Mac: follow installation instructions
- Windows:
  - Install docker desktop
  - Update wsl (in a command prompt): wsl – update

6.2.2 Create image for a project

When a project is ready to share, you can create a self-contained docker image:

```
# Clean environment ------------------------------------------------------

  # DELETE ALL CACHE
  system("docker builder prune --all -f")

  # Clean renv cache libraries
  renv::clean()

  # Clean extra versions of libraries
  source("R/helper_functions_extra.R")
  clean_renv_cache()


# Create docker image ----------------------------------------------------

  PID = 999
  jsPsychHelpeR::create_docker_container(PID = PID)
```

6.2.3 Store image

You can create a tar file with the image or directly share it through dockerhub:

1. Store image creating a pid[PID].tar.zip file TO SHARE

```
PID = 999
system(paste0("docker save gorkang/jspsychhelper:pid", PID, " | zip > pid", PID, ".tar.z
```

2. Push image to Dockerhub

```
PID = 999
system(paste0("docker push gorkang/jspsychhelper:pid", PID))
```

6.2.4 Load image

You can load the image in your computer in two ways:

1. Using a pid[PID].tar.zip:

On linux:

```
PID = 999
utils::unzip(zipfile = paste0("pid", PID, ".tar.zip"), files = paste0("-"))
system(paste0("docker load --input -"))
```

On windows:

```
tar -xf pid999.tar.zip & docker load input -
```

2. Pull image from Dockerhub

```
PID = 999
system(paste0("docker pull gorkang/jspsychhelper:pid", PID))
```

6.2.5 Run container

Once the docker image is loaded in your system, you will be able to run the data preparation and analysis for your project inside a docker container, ensuring reproducibility. The output will be in `Downloads/jsPsychHelpeR[PID]/outputs` after a couple of minutes.

- Linux

```
# Make sure ~/Downloads/jsPsychHelpeR999 is empty
file.remove(list.files(paste0("~/Downloads/jsPsychHelpeR", PID, "/outputs"), recursive =

# Run docker
```

```
  system(paste0("docker run --rm -d --name pid", PID, " -v ~/Downloads/jsPsychHelpeR", PID
```

- Windows

```
docker run --rm -d --name jspsychhelper -v %USERPROFILE%\Downloads\jsPsychHelpeR\outputs:/hor
```

6.2.6 DEBUG Container

You can DEBUG a container with the following command:

```
docker run --rm -ti -v ~/Downloads/jsPsychHelpeR999/outputs:/home/project/jsPsychHelpeR/outpu
```

Inside the container, you can acces R and debug as you would locally.

```
# See last CMD line in Dockerfile_TEMPLATE:
# $ R
source('renv/activate.R')
invisible(lapply(list.files('./R', full.names = TRUE, pattern = '.R$'), source))
setup_folders(pid = 999, folder = '.')
targets::tar_destroy(ask = FALSE)
targets::tar_make()

# Check size folders
du -had1 renv/ | sort -h
du -had1 * | sort -h
```

6.3 Google Docs

We have a few Google Documents with information about available tasks, protocols, etc.

- All tasks
- List of protocols
- NEW tasks
- Checks specific tasks

6.4 Folders and how to work

We have two main locations, the Github jsPsychMaker project and the server.

Github jsPsychMaker project

- canonical_protocol:
  - machinery: last stable version
  - tasks: all available tasks
  - server: `protocols/999/`
- canonical_protocol_DEV
  - machinery: development version
  - tasks: all available tasks
  - server: `protocols/test/canonical_protocol_DEV/`
- canonical_protocol_clean
  - machinery: last stable version
  - tasks: Consent and Goodbye
  - server: `protocols/test/canonical_protocol_clean/`
- protocols_DEV
  - machinery: last stable version
  - should only contain tasks in development
  - server: `protocols/test/protocols_DEV/`

In `protocols_DEV` we prepare the new **protocolos**:

- Create a copy in `test/protocols_DEV` of `canonical_protocol_clean` and rename to the number of the new protocol, `test/protocols_DEV/NumberOfProtocol`
- Once the protocol is ready:
  - Copy protocol to root folder: `protocols/NumberOfProtocol`
  - ZIP subfolder and move zip to `protocols/test/protocols_DEV/OLD_TESTS/`
  - Delete folder `test/protocols_DEV/NumberOfProtocol`
  - If there are new tasks:
    - CHECK with: check_missing_prepare_TASK()
    - TEST with create_protocol_with_NEW_tasks.R
    - Copy tasks, images, videos, specific plugins, etc. to `protocols/999/`
    - TEST in canonical protocol `protocols/999/` just in case there is a weird interaction

6.5 Helper functions

There are a number of helper functions to make some of the jsPsychR admins tasks easier.

6.5.1 Check all protocols

For example, we can use `check_missing_prepare_TASK()` to:

- Download all the protocols (without data) to a local folder (`sync_protocols = TRUE`)
- Check the trialid's of all the tests are OK (`check_trialids = TRUE`)
- Check there are no duplicate short_name of tasks in the tareas jsPsychR and NUEVAS tareas
- Check which tasks do not have a prepare_TASK.R script
- Check tasks with no info on the tareas jsPsychR Google doc
- Check tasks with missing info on NUEVAS tareas

```
# Open jsPsychHelpeR RStudio project

  # Load check_missing_prepare_TASK() function
  # cli::cli_alert_info(getwd())

  WD = getwd()
  setwd("../../jsPsychHelpeR/")
  source("R/check_missing_prepare_TASK.R")
  # source("../../jsPsychHelpeR/R/check_missing_prepare_TASK.R")
  setwd(WD)

  # If sync_protocols = TRUE, will download to ../CSCN-server/protocols all the protocols
  DF_missing = check_missing_prepare_TASK(sync_protocols = FALSE,
                                          check_trialids = TRUE,
                                          delete_nonexistent = TRUE,
                                          check_new_task_tabs = TRUE,
                                          helper_folder = "../../jsPsychHelpeR",
                                          CSCN_server_folder = "../../CSCN-server/")

  # - Tasks with no prepare_TASK() script!
  # - Tasks NOT in Google Doc
  # - Check trialid's are OK
  DF_missing
```

```
DF_missing$DF_FINAL %>% tidyr::replace_na(list(missing_script = "",
                                               missing_googledoc = "",
                                               missing_task = ""))


# Tasks ready to create prepare_*.R script
DF_missing$DF_FINAL %>% filter(!is.na(missing_script) & is.na(missing_gdoc))
DF_missing$DF_FINAL %>% filter(!is.na(missing_script) | !is.na(missing_gdoc)) %>%
  filter(!task %in% c("DEMOGR24", "DEMOGRfondecyt2022E1", "ITC", "fauxPasEv")) %>%  # "M
  select(-matches("missing"), -Nombre, -Descripcion)
```

6.5.2 Create protocol with NEW tasks

With `create_protocol_with_NEW_tasks.R` we can create a protocol with the tasks for which we do not yet have a control snapshot (no .csv's in the 999.zip data).

This is a necessary step before the task can be moved to the canonical protocol.

The function `find_missing_tasks_in_999()` will read all the csv in `jsPsychHelpeR/data/999/999.zip` and depending on the value of the parameter `search_where` ("prepare_TASK" or "js"):

- all .js tasks in `jsPsychMaker/protocols_DEV/`
- all prepare_TASK.R in `jsPSychHelpeR/R_tasks/`

Comparing both sources, will look for tasks for which we do not have a .csv in the 999 protocol yet (remember the 999 protocol is the canonical_protocol in the server).

Then, it prepares a NEW_TASKS protocol using the tasks.js files found in the server (after downloading all the server protocols to `jsPsychR/CSCN-server/`). A couple important points:

- This is a bit tricky, as it will use all the tasks in the server that it can found, across all the protocols, and will select the newest one.
- Sometimes there are multiple copies, with different dates and sizes…
- It is important that the server is as clean as possible. With all the OLD non-updated protocols zipped.

To make sure the Github `jsPsychMaker/protocols_DEV/NEW_TASKS/` is up to date, `create_protocol_with_NEW_tasks.R` will UPLOAD `CSCN-server/.../NEW_TASKS` to the server, and then DOWNLOAD NEW_TASKS to `../jsPsychMaker/protocols_DEV/NEW_TASKS/`

6.5.3 Check canonical protocol DEV

With `000_CHECK_CANONICAL.R` we can check that the canonical protocol in development works and expected.

In the script you can:

- sync `canonical_protocol_DEV/` folder in jsPsychMaker to `999/` in the server
- launch 5 monkeys
- rename the csv files to a fixed date, etc.
- prepare data
- compare with snaphot (WIP)

7 New protocols and tasks

There are a number of elements the tasks need to work well with jsPsychR, so we recommend to use one of the systems we have developed.

For example, with `jsPsychMaker::create_protocol()`, you can use tasks we already developed, and/or create new tasks defining their parameters in csv/excel files. The tasks will be part of a fully working protocol. You will need R 4.2 or higher to use it.

`create_protocol()` can:

- Loop through the subfolders in `folder_tasks` to create one task per subfolder
- Copy `canonical_protocol_clean` to `folder_output`
- Include in the protocol any tasks in `canonical_tasks`
- Modify `config.js` to add all the tasks created and selected
- Modify `index.html` to inlude only the plugins those tasks will use
- Modify `index.html` to add the media those task will use
- Check task names are OK: no spaces, -, _, do not start by a number, …
- Check names of all trialid's are OK (NAMETASK_NUMBER, e.g. MYTASK_001)
- Check only one csv or xls/xlsx file per folder
- Check plugins used exist in jsPsych-6/plugins
- Check we have the necessary parameters (WIP)
- Delete files of plugins not used

`create_protocol()` cannot yet:

- Modify config.js to adapt `all_conditions` to the experimental tasks added
- Use shiny app to edit the local config.js

7.1 New protocols

You can create a new protocol in seconds, choosing from the tasks we already have available.

Make sure you have the last version of jsPsychMaker, installing from Github:

```
if (!require('remotes')) install.packages('remotes'); library('remotes')
remotes::install_github("gorkang/jsPsychMaker")
```

Check if there are new tasks available in a new version of the Github package:

```
jsPsychMaker::check_NEW_tasks_Github()
```

7.1.1 List available tasks

You can list available tasks to choose from. You have more details in the section available-tasks.

```
  jsPsychMaker::list_available_tasks()
```

```
$tasks
 [1] "AIM"          "Bank"         "BART"         "BNT"          "bRCOPE"
 [6] "CAS"          "Consent"      "ConsentHTML"  "Cov19Q"       "COVIDCONTROL"
[11] "CRS"          "CRT7"         "CRTMCQ4"      "CRTv"         "DASS21"
[16] "DEBRIEF"      "DEMOGR"       "EAR"          "EmpaTom"      "ERQ"
[21] "ESM"          "fauxPasEv"    "GBS"          "GHQ12"        "Goodbye"
[26] "HRPVB"        "HRPVBpost"    "IBT"          "IDQ"          "IEC"
[31] "INFCONS"      "IRI"          "IRS"          "MDDF"         "MDMQ"
[36] "MIS"          "OBJNUM"       "OTRASRELIG"   "PBS"          "PRFBM"
[41] "PRFBMpost"    "PSETPP"       "PSPPC"        "PSS"          "PVC"
[46] "PWb"          "REI40"        "Report"       "RSS"          "RTS"
[51] "SASS"         "SBS"          "SCSORF"       "SDG"          "SRA"
[56] "SRBQP"        "SRSav"        "STAI"         "SWBQ"         "WEBEXEC"
```

```
$tasks_js
 [1] "AIM.js"          "Bank.js"         "BART.js"         "BNT.js"
 [5] "bRCOPE.js"       "CAS.js"          "Consent.js"      "ConsentHTML.js"
 [9] "Cov19Q.js"       "COVIDCONTROL.js" "CRS.js"          "CRT7.js"
[13] "CRTMCQ4.js"      "CRTv.js"         "DASS21.js"       "DEBRIEF.js"
[17] "DEMOGR.js"       "EAR.js"          "EmpaTom.js"      "ERQ.js"
[21] "ESM.js"          "fauxPasEv.js"    "GBS.js"          "GHQ12.js"
[25] "Goodbye.js"      "HRPVB.js"        "HRPVBpost.js"    "IBT.js"
[29] "IDQ.js"          "IEC.js"          "INFCONS.js"      "IRI.js"
[33] "IRS.js"          "MDDF.js"         "MDMQ.js"         "MIS.js"
[37] "OBJNUM.js"       "OTRASRELIG.js"   "PBS.js"          "PRFBM.js"
[41] "PRFBMpost.js"    "PSETPP.js"       "PSPPC.js"        "PSS.js"
[45] "PVC.js"          "PWb.js"          "REI40.js"        "Report.js"
[49] "RSS.js"          "RTS.js"          "SASS.js"         "SBS.js"
[53] "SCSORF.js"       "SDG.js"          "SRA.js"          "SRBQP.js"
[57] "SRSav.js"        "STAI.js"         "SWBQ.js"         "WEBEXEC.js"
```

7.1.2 Create a protocol

This will create a fully working protocol in `folder_output`. You can edit `config.js` to adapt the protocol to your needs. See experiment configuration for more details.

```
jsPsychMaker::create_protocol(canonical_tasks = c("AIM", "EAR", "IRI"),
                              folder_output = "~/Downloads/TEST/new_protocol",
                              launch_browser = TRUE)
```

7.2 New tasks

7.2.1 Create tasks

You can create new tasks with `create_task()` using `csv` or `xls/xlxs` files for the items, and `html` files for the instructions. But we recommend you use `create_protocol()` instead, so the tasks will be part of a fully working protocol, and testing them will be a breeze.

There are some things to take into account:

- **folder_tasks** expects a folder with sub-folders with the ShortName of tasks. Inside, they need to have one `csv` or `xls/xlxs` file and `html` files. Use `jsPsychMaker::copy_example_tasks(destinat = "~/Downloads/TEST")` to see a working example

- The **csv or xls/xlsx file** (ShortName.csv or Shortname.xls/xlsx) needs to have an `ID` and `plugin` columns, and then columns by the name of parameters used in the plugin (e.g. if using the `survey-text` plugin, you will need the `prompt` parameter). If you need **help with the plugins parameters**, see the jsPsych 6.3 list of plugins
- For each **html file** (ShortName_instructions.html, ShortName_instructions2.html, etc) an instructions page will be created. The files need to end with `_instructions.html` or `instructions#.html` (# is a number). If there is no html, a default page will be used
- For **key questions** (e.g. present this question only if participants responded "3"), you need to create a column named `if_question` and include a logical condition. For example:
  - `1 != 25`: Response to item 1 is NOT 25
  - `3 == 20`: Response to item 3 is 20
  - `15 == yes`: Response to item 3 is yes
- If you use tasks with images, video or audio, make sure to include the files in a `media/` folder:
  - Images: `media/img`
  - Videos: `media/vid`
  - Audio: `media/audio`
- If you use a plugin with different options or alternatives (e.g. `survey-multi-choice-vertical`), the different response `options` will be the words or sentences separated by commas (e.g. Yes I am, No I am not). If your options have commas (e.g. Wake up, have breakfast, brush teeth; Wake up, brush teeth, have breakfast), you need to define an `options_separator` using the `options_separator = ";"` parameter.

You can run the fully reproducible example included in jsPsychMaker:

1) Install jsPsychMaker from Github and load library

```
if (!require('remotes')) install.packages('remotes'); library('remotes')
if (!require('jsPsychMaker')) remotes::install_github("gorkang/jsPsychMaker"); library('js
```

2) Copy example tasks

This will copy a few example tasks that you use to adapt your tasks. For example, `MultiChoice` and `Slider` tasks, a key questions mini-task (`IfQuestion`), and an `ImageButtonResponse` task.

```
jsPsychMaker::copy_example_tasks(destination_folder = "~/Downloads/TEST")
```

3) Create your protocol

```
# Create protocol
jsPsychMaker::create_protocol(folder_tasks = "~/Downloads/TEST/",
                              folder_output = "~/Downloads/TEST/new_protocol",
                              launch_browser = TRUE)
```

7.3 HELP with new tasks

If you need help developing new tasks, you can open a new Issue in the jsPsychMaker Github.

We will ask you to add the details about the task in the NEW tasks document.

Once the task is implemented, our goal is to always end up having a sister task preparation script in jsPsychHelpeR. You can try to create the preparation script and do a Pull request, or ask for help opening a new Issue in the jsPsychHelpeR Github.

7.3.1 How to fill the NEW tasks document

NEW tasks document

First of all, you will need the original paper where the task was validated/translated to have all the details at hand. Please, send us a link to the paper.

The best way to fill the NEW tasks document is:

1. Find a task similar to yours in the document Tareas jsPsychR where we have information about all the available tasks.
2. Copy/paste the information from all the tabs to the NEW tasks document and adapt it.

Try to be as consistent as possible. For example, when entering the information about numeric conversion in the Puntajes_items tab:

```
All the cells must be:
1 = Mucho
2 = Poco
...
```

DO NOT do things like:

```
1: Mucho
1 Mucho
1 pto = Mucho
Mucho 1
```

Please, make sure you fill out all the details in all the tabs.

8 CreateSimulatePrepare

Here you can see the full process of creating a protocol with jsPsychMaker, simulating participants with jsPsychMonkeys and preparing the data with jsPsychHelpeR:

8.1 Create protocol

Create a protocol with `jsPsychMaker::create_protocol()`:

```
# 1) Install jsPsychMaker
if (!require('remotes')) install.packages('remotes'); remotes::install_github("gorkang/jsP

# 2) Check available tasks
jsPsychMaker::list_available_tasks()$tasks

# 3) Create protocol
jsPsychMaker::create_protocol(canonical_tasks = c("AIM", "EAR", "IRI"),
                              folder_output = "~/Downloads/protocol999",
                              launch_browser = FALSE)
```

You can now edit the configuration file (`~/Downloads/protocol999/config.js`) to adjust the project's parameters.

8.2 Simulate participants

Simulate participants with {jsPsychMonkeys}. Make sure your system has a functioning docker installation, see jsPsychMonkey's setup:

```
# 1) Install jsPsychMonkeys
if (!require('remotes')) utils::install.packages('remotes'); remotes::install_github('gork

# 2) Run monkeys
  # Go to _targets.R: Change parameter `local_folder_tasks` to your folder_output above. F
  # - On Ubuntu, `local_folder_tasks = "~/Downloads/protocol999"`
```

```
  # - On Windows, `local_folder_tasks = "C:/Users/myusername/Downloads/protocol999"`

jsPsychMonkeys::release_the_monkeys(uid = 1:10,
                                    local_folder_tasks = "~/Downloads/protocol999/")
```

The monkeys responses csv's should be initially downloaded in your `Downloads` folder, and automatically moved to a `.data/` folder inside the protocol folder. For example, `~/Downloads/protocol999/.data`

8.3 Prepare data

Create a data preparation project with `jsPsychHelpeR::run_initial_setup()`:

```
# 1) Install
if (!require('remotes')) install.packages('remotes'); remotes::install_github("gorkang/jsP

# 2) Create project
jsPsychHelpeR::run_initial_setup(pid = "999", data_location = "~/Downloads/protocol999/.da

# 3) Restore all the necessary packages using {renv}
renv::restore(prompt = FALSE)

# 4) Run data preparation
targets::tar_make()
```

If you don't give a value to the `folder` parameter in `jsPsychHelpeR::run_initial_setup()`, the new project will be created in `~/Downloads/jsPsychHelpeRtest/`. After step 4), the prepared data can be found in the `outputs/data` folder of the new project, reports in `outputs/reports`, etc.

9 Common Tasks

9.1 Install dependencies

```
# if (!require('renv')) utils::install.packages('renv'); renv::install("gorkang/jsPsychAdm
```

9.2 Developing a protocol

9.2.1 Creating a protocol

```
# if (!require('renv')) utils::install.packages('renv'); renv::install("gorkang/jsPsychMak
```

9.2.2 Adding new tasks

```
# TODO
```

9.2.3 Piloting the protocol

```
# if (!require('renv')) utils::install.packages('renv'); renv::install('gorkang/jsPsychMon

jsPsychMonkeys::release_the_monkeys(uid = "1:100",
                    sequential_parallel = "parallel",
                    number_of_cores = 10,
                    server_folder_tasks = "test/protocols_DEV/999",
                    DEBUG = FALSE,
                    credentials_folder = "/home/emrys/gorkang@gmail.com/RESEARCH/PROYECTOS
                    open_VNC = FALSE)
```

9.2.4 Deleting pilot data

```
  # Delete the XYZ protocol rows in all the MYSQL tables
  source("admin/mysql_helper_functions.R")
  # list_credentials = decrypt_data(key_public = readLines(".vault/data_public_key.txt")
  delete_MySQL_tables_pid(pid)


# 3) Limpiar los archivos de resultados de Monkeys ----------------------

  # Delete csv files in .data/
  # rstudioapi::navigateToFile(".vault/.credentials")
  DELETE_data_server(pid = PROTOCOLID)
```

9.2.5 Preparing Helper project

9.2.5.1 Get data

Will download a zip file with the data.

```
# if (!require('renv')) utils::install.packages('renv'); renv::install('gorkang/jsPsychHel

# Developing protocol
jsPsychHelpeR::get_zip(pid = "test/protocols_DEV/31", what = "data",  where = "data/")

# Production protocol
jsPsychHelpeR::get_zip(pid = "999", what = "data",  where = "data/")
```

9.2.6 Prepare new task correction project

```
  jsPsychHelpeR::run_initial_setup(pid = 999,
                                   download_files = TRUE,
                                   folder = "~/Downloads/jsPsychR_MYPROJECT")
```

9.3 Protocol to production

From `admin/000_PREPARE_protocol_for_production.R` in jsPsychAdmin

```r
# Checklist para pasar protocolos de test/protocols_DEV/ a produccion

  # PARAMETERS -------------------------------

  PROTOCOLID = "test/protocols_DEV/31"
  number_of_monkeys = "1:100"

  # ------------------------------------------

  # Automatic parameters
  pid = gsub("test/protocols_DEV/", "", PROTOCOLID)


   cli::cli_h1("PROTOCOL {pid}")


  # 1) Pilotaje final on Monkeys! ------------------------------------------

  # Clean data and MySQL DB
  # rstudioapi::navigateToFile(".vault/.credentials")
  jsPsychAdmin::clean_up_dev_protocol(protocol_id = PROTOCOLID) # Will ask for server pass


  # LAUNCH MONKEYS
  jsPsychMonkeys::release_the_monkeys(uid = number_of_monkeys,
                                      server_folder_tasks = PROTOCOLID,
                                      sequential_parallel = "parallel",
                                      number_of_cores = 10,
                                      big_container = TRUE,
                                      keep_alive = FALSE, open_VNC = FALSE, screenshot = F
                                      credentials_folder = here::here(".vault/"))


  # CHECK jsPsychHelpeR runs OK

  # || THIS WILL take a while, as all the renv packages need to update
  # Create NEW jsPsychHelpeR project, downloading the files from the server
  jsPsychHelpeR::run_initial_setup(pid = PROTOCOLID,
                                   download_files = TRUE,
                                   folder = "~/Downloads/jsPsychR_TESTING_for_PRODUCTION")
```

```
  # REMEMBER TO DO targets::tar_make() in jsPsychHelpeR project!


# 2) Clean data and  MySQL DB ------------------------------------------

  # rstudioapi::navigateToFile(".vault/.credentials")
  jsPsychAdmin::clean_up_dev_protocol(protocol_id = pid) # Will ask for server password


# 3) Revisar el config.js para pasar el experiment a produccion ----------

  # -[] online = true
  # -[] pid OK?
  # -[] debug_mode = false
  # - ETC...


# 4) Copiar protocolo ZIPeado a test/protocols_DEV/OLD_TESTS/ -------------

  # TODO: automatico!

# 5) Copiar protocolo a protocols/ ---------------------------------------

  # TODO: automatico!

# 6) BORRAR protocolo de test/protocols_DEV/OLD_TESTS/ -------------------

  # TODO: automatico!
```

References

- jsPsych
- targets

Landau, William Michael. 2021. “The Targets r Package: A Dynamic Make-Like Function-Oriented Pipeline Toolkit for Reproducibility and High-Performance Computing” 6: 2959. https://doi.org/10.21105/joss.02959.