

# Getting Started

*David M. Freestone, Mara Vinnik, & Cherie Doan*

*March 25, 2017*

## Introduction

[Have to write this]

Note that the folder structure and all the files mentioned below are freely available on our github page.

## Folders & Files

A Dropbox account is **highly** recommended. In our experience, Google Drive is slow, cpu intensive when synching, and fails to sync more often than Dropbox. Using a remote server is fine, except that in our experience it is a hassle to make sure you are always connected to it from any computer you're likely to be working on. In what follows, we will assume you will use dropbox, but if not, you can replace all mention of dropbox with the top level folder you are using.

The dropbox folder should have at least 2 folders in it: *Active* and *system*. Our dropbox account has 4, and is shown below.

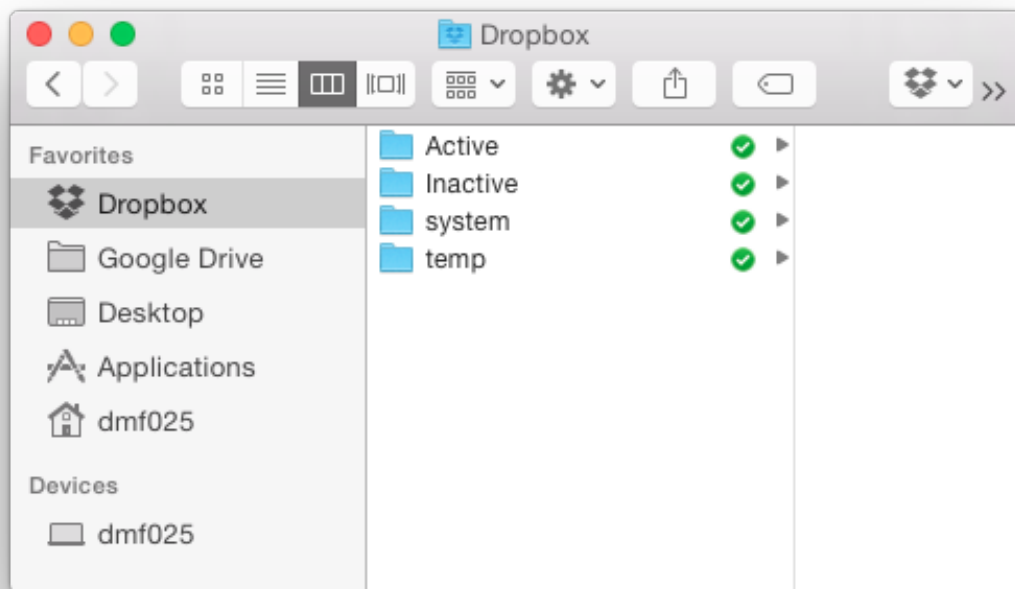


Figure 1: Figure 1. An example Dropbox Folder

## The *Active* Folder

The Active folder contains settings specific to the currently active experiments. At a minimum, it should contain *macros*, *figures*, and the folder for any currently active experiments. If you would like email notifications, you must also include a text file named *emails.txt* that contains the email addresses of those that should receive email notifications, one email per line.

The *figures* folder contains the figures that are automatically generated by the daily analyses scripts that are run.

The *macros* folder contains, in our case, the macros for the two cabinets that house the individual rodent chamber. It also contains an additional folder, *submacs* that contain the settings for each individual rodent chamber. These macros can be automatically generated, as described later.

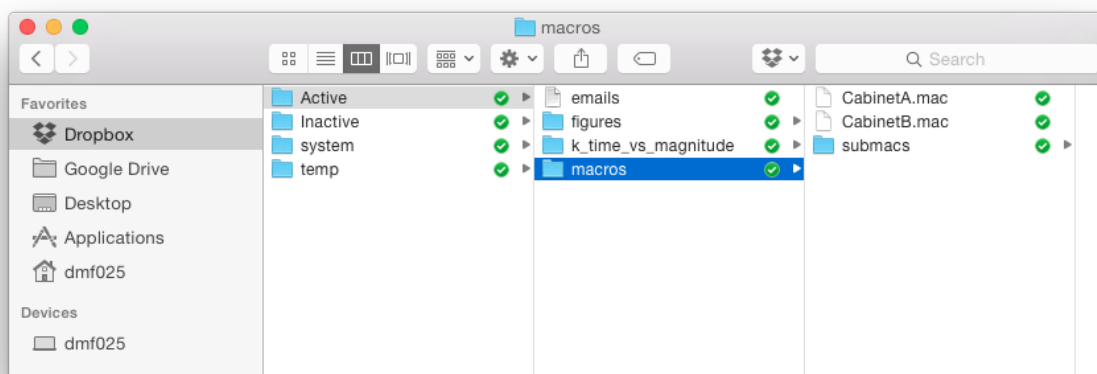


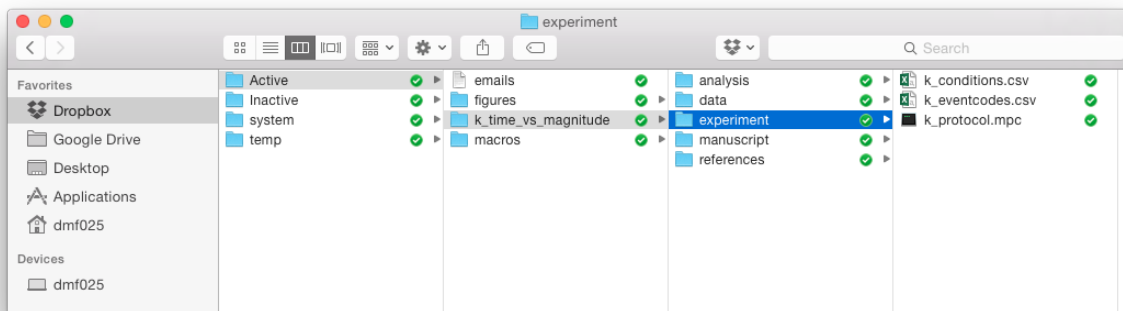
Figure 2: Figure 2. An example macros Folder

Every experiment should be placed in its own separate folder. The folder must begin with the experiment ID followed by an underscore. This requirement is in place because the software ensures all aspects of the experiment are valid and connected, and it does this using the unique experiment identifier. In the example shown below, the active experiment has experiment ID *k* and is named *time\_vs\_magnitude*.

The folder structure of each experiment data file must be nearly identical. This is because the software assumes the location of several key files, like the protocol that runs the MPC code. The experiment folder must have, at a minimum, *data* and *experiment* folders.

The *data* folder contains two additional folders, *mpc* and *json*. The *mpc* folder is where the mpc files are saved to disk in real-time by the MedPC software. The *json* folder is where the daily .JSON files are kept. These JSON files contain the raw data from each mpc file each day, but also contains the protocol, conditions, and event codes that were in place that day. This JSON file is a key component of our system as it allows you to keep your data and the metadata in a single place. This file format was chosen because of its widespread use in computer science. It is a standard format for storing hierarchical data. In the fully automated system, a single JSON file is created automatically each day. This ensures that the metadata is tied to the data as soon as possible. By storing the data in separate JSON files, we maximize the information contained in each file per day, and minimize the fallout from having lost a data file (as would happen if all the data were stored in a single file).

The *experiment* folder must contain at a minimum *ID\_protocol.mpc*, *ID\_conditions.csv*, and *ID\_eventcodes.csv*. You should replace *ID* with your unique experiment ID. These files control the experiment. A picture of our current experiment folder is shown below.



The *k\_protocol.mpc* file contains the the actual mpc code that runs the operant chambers.

The *k\_conditions.csv* file contains the conditions for every rat in the experiment.

The *k\_eventcodes.csv* file contains the eventcodes specific to your particular experiment (a more general event codes file specific to the entire system is in the *system* folder, described later)

These files must be kept up-to-date because these are joined with the data per day in the JSON file.

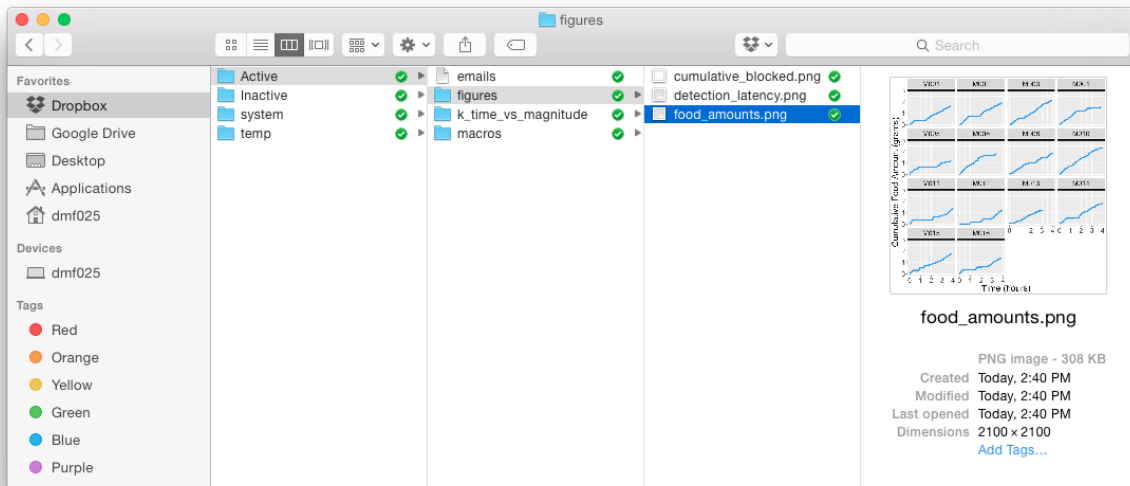
The image above shows several additional folders we use in our lab for each experiment: *analyses* stores the analysis code (we call the entry point to the code *k\_main.R*), the *manuscript* folder (which contains constantly updating drafts of the manuscript), and a *references* folder that contains PDFs of the references particularly relevant for this experiment.

### The *system* folder

Zooming back out, the Dropbox folder must also contain a folder called *system*. This folder contains files that are general to the entire running system.

The *mouse\_eventcodes.csv* file contains the eventcodes that are general to the system. This includes every nosepoke, or the onsets and terminations of hopper lights, the delivery of food pellets, the detection of those pellets, etc. All eventcodes specific to an experiment (for example, a trial start code) should be kept in the experiment specific event codes file described above (e.g., *k\_eventcodes.csv* in the *experiment* folder).

The *boxtest.R* file runs the automated boxtests to check for a functioning system. For example, it checks the latency between the delivery of a food pellet and when it finally is detected in the food cup (a long delay can indicate a malfunctioning feeder). Another example is to check the amount of food consumed in a day. It should generate figures that are put in *Active/figures* folder. An example is shown below.



The *archive.R* file reads in the daily mpc files and converts them to JSON.

The *macros.R* file reads in the *ID\_conditions.csv* file and automatically creates the macros in the *Active/macros* folder.

These R files are designed to be run from the command line, although they can be run in R or RStudio as well. Because the code can be run from the command line, it is easy to have them run based on a operating system specific scheduler. For example, below is the crontab schedule for our system. It shows 3 scheduled jobs. At 2:40pm (“40 14 \* \* \*”) and 10:40pm the boxtest.R file is run and the output is logged. (These times were chosen because they are 10 minutes after the active feeding period in the experiment). And a third job runs and logs the *archive.R* file at 12:30am (30 minutes after the date changed).

```
1. vim
## cronR job
## id: job_f2c8b639ead8db12b4dea5971de78e4
## tags:
## desc: Do a box test.
40 14 * * * /Library/Frameworks/R.framework/Resources/bin/Rscript /Users/dmf025/Dropbox/system/boxtest.R >> /Users/dmf025/Dropbox/system/boxtest.log 2>&1

## cronR job
## id: job_120630de825241a80cf834d7ac9c17ee
## tags:
## desc: Nightly boxtest
40 22 * * * /Library/Frameworks/R.framework/Resources/bin/Rscript /Users/dmf025/Dropbox/system/boxtest.R >> /Users/dmf025/Dropbox/system/boxtest.log 2>&1

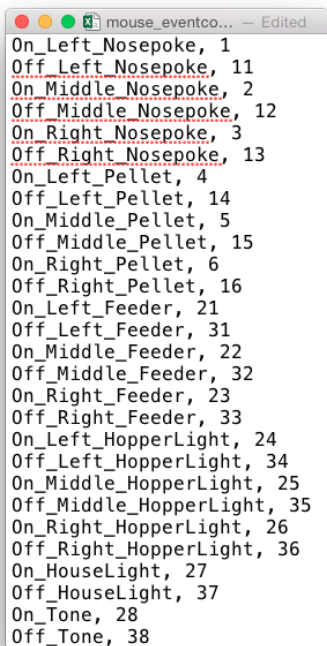
## cronR job
## id: job_5b704ee0b5533adc641794f1e3eb5e52
## tags:
## desc: Archive the day's event.
30 0 * * * /Library/Frameworks/R.framework/Resources/bin/Rscript /Users/dmf025/Dropbox/system/archive.R >> /Users/dmf025/Dropbox/system/archive.log 2>&1
```

Figure 3: Figure 5. An example schedule

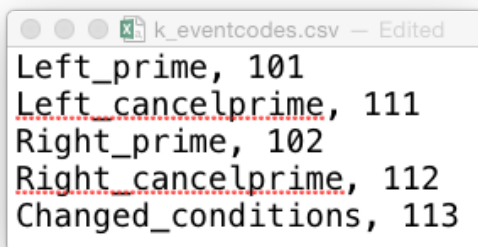
Lastly, our setup has an *Inactive* folder for completed projects, and a *temp* folder for temporary files. When an experiment is finished, we simply move it from *Active* to *Inactive*. This is important because the automated system will look for and identify any active experiments by looking in the *Active* folder. It alerts you to any conflicts (for example, if different protocols are to be run for the same mouse), and this prevents you from storing completed experiments in the *Active* folder.

## Eventcodes

There are two eventcodes files, the general one is stored in the *system* folder, and the experiment-specific one is stored in the *Active/ID\_experiment/experiment* folder, as described above. Both of these files are simply csv files that store eventname, eventcode pairs. For example, here are a snapshot of the *mouse\_eventcodes.csv* and *k\_eventcodes.csv* csv files.



```
On_Left_Nosepoke, 1
Off_Left_Nosepoke, 11
On_Middle_Nosepoke, 2
Off_Middle_Nosepoke, 12
On_Right_Nosepoke, 3
Off_Right_Nosepoke, 13
On_Left_Pellet, 4
Off_Left_Pellet, 14
On_Middle_Pellet, 5
Off_Middle_Pellet, 15
On_Right_Pellet, 6
Off_Right_Pellet, 16
On_Left_Feeder, 21
Off_Left_Feeder, 31
On_Middle_Feeder, 22
Off_Middle_Feeder, 32
On_Right_Feeder, 23
Off_Right_Feeder, 33
On_Left_HopperLight, 24
Off_Left_HopperLight, 34
On_Middle_HopperLight, 25
Off_Middle_HopperLight, 35
On_Right_HopperLight, 26
Off_Right_HopperLight, 36
On_HouseLight, 27
Off_HouseLight, 37
On_Tone, 28
Off_Tone, 38
```



```
Left_prime, 101
Left_cancelprime, 111
Right_prime, 102
Right_cancelprime, 112
Changed_conditions, 113
```

The data itself is stored using the event code. This allows us to store the data directly from MedPC as time.event pairs. The number before the period in the raw data is the MedPC timestamp, and the number after the period is the event code. For example, a line in the MedPC file that reads 16735.021 is parsed at the period. The timestamp 16735 corresponds to 167.35 seconds since the protocol was loaded, and the event code 21 corresponds to an On\_Left\_Feeder event. That is, a pellet was delivered into the left hopper 167.35 seconds after the protocol was loaded.

The eventcodes file forms a mapping between the event and the code in a way that the system can read in and match quickly and easily. The software component of the system knows to transform the event codes into event names when the files are loaded into memory for analysis. This allows the user to use the event name (e.g., “On\_Left\_Feeder” instead of the event code 21) in their analyses. Both the general and

experiment-specific eventcodes files are stored with the data in the JSON file every day. This compromise allows us to store data efficiently, but analyze that data using human-readable events.

## Conditions

A conditions file stores the conditions of the experiment for every mouse in the experiment. This conditions file is primarily used to automatically generate the macros used by MedPC. It is stored each day along with the data in the JSON file. For now, it is the user's responsibility to make sure that the MedPC program is using the latest macros and conditions file. But we will likely make this automatic in the future. The conditions file is stored in the *experiment* folder as *ID\_conditions.csv*. Here is a part of the conditions file for the *k* experiment. The picture is rendered as a table, but importantly, the file itself is a csv file.

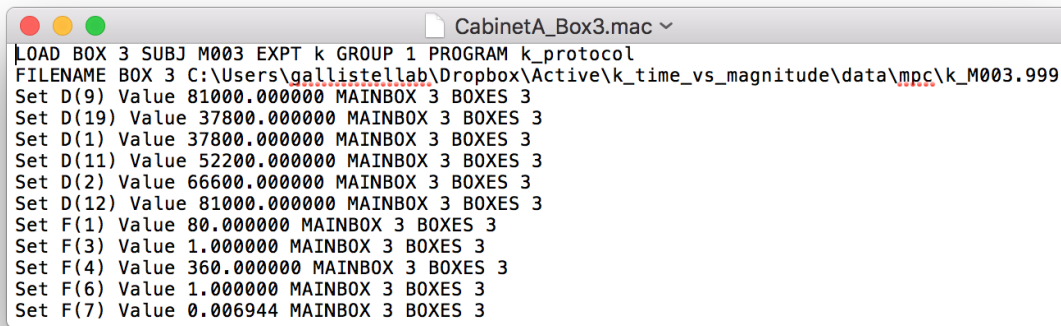
	A	B	C	D	E	F	G	H	I	J	K	L
1											On_Daytime (1030pm)	Off_Daytime (1030am)
2	Lab	Species	Strain	Sex	Supplier	Weights	Mouse	Cabinet	Box	Protocol	D(9)	D(19)
3	Freestone	Mouse	C57BL/6	Female	JAX		M001	A		1 k_protocol	81000	37800
4	Freestone	Mouse	C57BL/6	Female	JAX		M002	A		2 k_protocol	81000	37800
5	Freestone	Mouse	C57BL/6	Female	JAX		M003	A		3 k_protocol	81000	37800
6	Freestone	Mouse	C57BL/6	Female	JAX		M004	A		4 k_protocol	81000	37800
7	Freestone	Mouse	C57BL/6	Female	JAX		M005	A		5 k_protocol	81000	37800
8	Freestone	Mouse	C57BL/6	Female	JAX		M006	A		6 k_protocol	81000	37800
9	Freestone	Mouse	C57BL/6	Female	JAX		M008	A		8 k_protocol	81000	37800
10	Freestone	Mouse	C57BL/6	Male	JAX		M010	B		2 k_protocol	81000	37800
11	Freestone	Mouse	C57BL/6	Male	JAX		M011	B		3 k_protocol	81000	37800
12	Freestone	Mouse	C57BL/6	Male	JAX		M012	B		4 k_protocol	81000	37800
13	Freestone	Mouse	C57BL/6	Male	JAX		M013	B		5 k_protocol	81000	37800
14	Freestone	Mouse	C57BL/6	Male	JAX		M014	B		6 k_protocol	81000	37800
15	Freestone	Mouse	C57BL/6	Male	JAX		M015	B		7 k_protocol	81000	37800
16	Freestone	Mouse	C57BL/6	Male	JAX		M016	B		8 k_protocol	81000	37800

Figure 4: Figure 7. An example experiment conditions file

The first part of the conditions file stores information about the experiment itself, i.e., the lab in which it was run, the species, strain, sex of the animal, the supplier, the weights, and the unique mouse ID tag. The next two give the cabinet and box number in which the protocol (given in the Protocol column) will run. The following columns are values specific to the current settings of the experiment. In the MedPC template (discussed below), we use the array *D* to store anything associated with daily time operations; this includes the day/night cycle, the time of the feeding periods, etc. These are given in seconds since midnight (because they are checked in MedPC that way). Here, *D(9)* stores the time that the houselight should turn on (81000 seconds from midnight is 10:30pm). And *D(19)* stores the time that the houselight should turn off (378000 seconds from midnight is 10:30am). The first row of the conditions file is reserved for a human-readable explanation of the column, and is not used to generate the macros. For readability, this snapshot does not show all the variables set in the conditions file. But here is an example of a submac (a macro for a specific mouse) generated from this file. In the MedPC template, the *F* array is used to store any experiment specific variables.

## MedPC

This system is designed to be modular, but it works best if all the modules follow the same conventions. You can program your MedPC code in anyway you are comfortable with, but we provide a template designed for our system, called *0\_protocol.mpc*. This template takes care of all the boiler plate operations for you. It automatically records any event related to the normal functioning of the system. These include any input coming from the box (e.g., nosepokes from the mouse, a pellet detected in the trough), and any output changing the state of the box (e.g., a food delivery or a light onset or termination). For output, this is accomplished by replacing your standard operations (e.g., ON1) with specialized z-pulses (e.g., z^On\_Left\_Feeder) in your code. The template also automatically takes care of the day/night cycle and any feeding periods that you specify in your conditions file.



```
LOAD BOX 3 SUBJ M003 EXPT k GROUP 1 PROGRAM k_protocol
FILENAME BOX 3 C:\Users\gallistellab\Dropbox\Active\k_time_vs_magnitude\data\mpc\k_M003.999
Set D(9) Value 81000.000000 MAINBOX 3 BOXES 3
Set D(19) Value 37800.000000 MAINBOX 3 BOXES 3
Set D(1) Value 37800.000000 MAINBOX 3 BOXES 3
Set D(11) Value 52200.000000 MAINBOX 3 BOXES 3
Set D(2) Value 66600.000000 MAINBOX 3 BOXES 3
Set D(12) Value 81000.000000 MAINBOX 3 BOXES 3
Set F(1) Value 80.000000 MAINBOX 3 BOXES 3
Set F(3) Value 1.000000 MAINBOX 3 BOXES 3
Set F(4) Value 360.000000 MAINBOX 3 BOXES 3
Set F(6) Value 1.000000 MAINBOX 3 BOXES 3
Set F(7) Value 0.006944 MAINBOX 3 BOXES 3
```

Figure 5: Figure 8. An example submacro for a mouse

These operations take up 17 state sets. The 18th state set is for your SHOW commands, and the remaining 14 state sets are available for your experiment. (In our experience, even the most verbose experiment does not take up 14 state sets). Again, these state sets can be programmed in any way you choose, but you get the most out of plugging in to the existing template API. For example, we recommend using constants as indices into a single F array. This allows you to use human-readable variable names, and minimizes the number of single-letter variable names used. Another example was already mentioned: using the drop-in replacements for sending output to the chambers (e.g., `z^On_Left_Feeder`).

The data is automatically recorded and stored for you. The current version stores the data on disk under three conditions: First, at the end of every feeding period; Second, if the user issues a K-23 pulse manually; and Third, if the mouse/protocol has generated many events (the number of events needed to trigger this is set by the size of the Z array in the MedPC template).

The data is automatically stored as .999 file. This code tells the system that the data has not been archived, and contains the most recent data.