Analyse your behavioural data with `rethomics`

Quentin Geissmann
quentin.geissmann13@imperial.ac.uk

June 17, 2015

1 Installation

rethomics is still under heavy development, so it cannot be uploaded to the CRAN (Comprehensive R Archive Network) – where most stable packages are – yet. Instead, we can install R using Hadley Wickham's popular `devtools`[1]. Once you have `devtools` installed, it should be straightforward to install `rethomics`

```
library(devtools)
install_github("gilestrolab/rethomics",subdir = "rethomics")
```

Check for error messages. Then, ensure the package is installed by loading it:

```
library(rethomics)

## Loading required package:  MASS
## Loading required package:  RSQLite
## Loading required package:  DBI
## Loading required package:  data.table
## Loading required package:  pracma
## Loading required package:  zoo
##
## Attaching package:  'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
## Loading required package:  ggplot2
```

2 Loading data

2.1 Data structure

In `rethomics` the goal is to store *all* behavioural data in one single 'data table' – which is standard – for subsequent statistical analysis. In such a data table, every row correspond to a single measurement; that is the position of one animal at one time. Every column describes a statistical variable such as t (i.e. time in second), and X and Y positions, but it can also hold information about arbitrary conditions such as treatment, sex, age, genotype and so on.

By convention:

- t, the time of a measurment, is always *in seconds*
- X and Y, are *relative to the width of the region* they come from, and the *origin is top-left*

  For a single **experiment**, and when tracking a single animal (i.e. in a single **region**), your data table could look like:

```
data(multiple_iterative_y_mazes)
single_animal <- multiple_iterative_y_mazes[
                experiment_id == "female_FALSE_11.db"]
print(single_animal)

##             experiment_id region_id    sex sleep_deprived id               path       t
##    1: female_FALSE_11.db         1 female          FALSE 11 female_FALSE_11.db  12.166
##    2: female_FALSE_11.db         1 female          FALSE 11 female_FALSE_11.db  12.200
##    3: female_FALSE_11.db         1 female          FALSE 11 female_FALSE_11.db  12.233
##    4: female_FALSE_11.db         1 female          FALSE 11 female_FALSE_11.db  12.266
##    5: female_FALSE_11.db         1 female          FALSE 11 female_FALSE_11.db  12.300
```

[1]installation instructions are available here

```
##   ---
## 4167: female_FALSE_11.db         1 female          FALSE 11 female_FALSE_11.db 151.633
## 4168: female_FALSE_11.db         1 female          FALSE 11 female_FALSE_11.db 151.666
## 4169: female_FALSE_11.db         1 female          FALSE 11 female_FALSE_11.db 151.833
## 4170: female_FALSE_11.db         1 female          FALSE 11 female_FALSE_11.db 151.966
## 4171: female_FALSE_11.db         1 female          FALSE 11 female_FALSE_11.db 152.100
##               x           y           w           h phi has_interacted
##    1: 0.5505208 0.565104167 0.005729167 0.001562500   0          FALSE
##    2: 0.5510417 0.564062500 0.008854167 0.003645833   0          FALSE
##    3: 0.5520833 0.563020833 0.009375000 0.005729167   0          FALSE
##    4: 0.5515625 0.562500000 0.011458333 0.006770833   0          FALSE
##    5: 0.5510417 0.561979167 0.011458333 0.007812500   0          FALSE
##   ---
## 4167: 0.6270833 0.002604167 0.009375000 0.003645833   0          FALSE
## 4168: 0.6265625 0.002604167 0.007812500 0.003645833   0          FALSE
## 4169: 0.6369792 0.001562500 0.003645833 0.001562500   0          FALSE
## 4170: 0.6328125 0.001562500 0.004687500 0.001562500   0          FALSE
## 4171: 0.6125000 0.003125000 0.004687500 0.004687500  90          FALSE
```

You may notice immediatly that the first few columns seem unnecessary as they have constant values. This is because they describe variables that vary between individuals, but we have only one individual in this simplistic examples.

From simply looking at this table, you should be able to answer:

- How many measurment have been made in this experiement?
- After how many seconds was the first measurment made?, and the last?
  In order to understand the need for additional columns, we can load data from multiple experiments:

  ```
  data(multiple_iterative_y_mazes)
  print(multiple_iterative_y_mazes)
  ```

  ```
  ##             experiment_id region_id    sex sleep_deprived id               path       t
  ##     1: female_FALSE_11.db         1 female          FALSE 11 female_FALSE_11.db  12.166
  ##     2: female_FALSE_11.db         1 female          FALSE 11 female_FALSE_11.db  12.200
  ##     3: female_FALSE_11.db         1 female          FALSE 11 female_FALSE_11.db  12.233
  ##     4: female_FALSE_11.db         1 female          FALSE 11 female_FALSE_11.db  12.266
  ##     5: female_FALSE_11.db         1 female          FALSE 11 female_FALSE_11.db  12.300
  ##    ---
  ## 40426:   male_FALSE_25.db         1   male          FALSE 25   male_FALSE_25.db 191.966
  ## 40427:   male_FALSE_25.db         1   male          FALSE 25   male_FALSE_25.db 192.000
  ## 40428:   male_FALSE_25.db         1   male          FALSE 25   male_FALSE_25.db 192.033
  ## 40429:   male_FALSE_25.db         1   male          FALSE 25   male_FALSE_25.db 192.066
  ## 40430:   male_FALSE_25.db         1   male          FALSE 25   male_FALSE_25.db 192.100
  ##                x           y           w           h phi has_interacted
  ##     1: 0.5505208 0.565104167 0.005729167 0.001562500   0          FALSE
  ##     2: 0.5510417 0.564062500 0.008854167 0.003645833   0          FALSE
  ##     3: 0.5520833 0.563020833 0.009375000 0.005729167   0          FALSE
  ##     4: 0.5515625 0.562500000 0.011458333 0.006770833   0          FALSE
  ##     5: 0.5510417 0.561979167 0.011458333 0.007812500   0          FALSE
  ##    ---
  ## 40426: 0.3661458 0.005208333 0.025520833 0.011458333  36          FALSE
  ## 40427: 0.3651042 0.004687500 0.024479167 0.011458333  37          FALSE
  ## 40428: 0.3645833 0.004166667 0.023958333 0.011979167  35          FALSE
  ## 40429: 0.3625000 0.006770833 0.021875000 0.011979167   0          FALSE
  ## 40430: 0.3625000 0.006250000 0.021875000 0.010937500   0          FALSE
  ```

  Any data data should *always* have two columns: **experiment_id** and **region_id**. Together, these columns constitue a so called *key*. In other words, any unique combination of experiment *and* region represents a single animal. As a result, we can identify, unambiguously, any animal given we know its region and which experiement it comes from. Instead of using the legacy data.frames,

`rethomic` takes advantage of Matt Doyle's powerful `data.table` package[2]. This makes it very easy and efficient to work with large amount of behavioural data. Common operation could involve filtering data and computing variable per condition or per individual.
Let us go through several examples:

```
data(multiple_iterative_y_mazes)
#We can simply call this data table `dt'
dt <- multiple_iterative_y_mazes
# keeping only females
dt_female <- dt[sex == 'female',]
# excluding any data point before 30 seconds (i.e. keeping >= 30s)
dt_currated <- dt[t >= 30,]
# Computing, per animal, the time spent in the experiment
summary_dt =  dt[,
                 .(time_spent = max(t) - min(t)),
                 by=key(dt)]
print(summary_dt)
```

```
##         experiment_id region_id time_spent
## 1: female_FALSE_11.db         1    139.934
## 2:  female_FALSE_2.db         1    149.300
## 3:  female_TRUE_11.db         1    235.200
## 4:  female_TRUE_21.db         1    147.866
## 5:   female_TRUE_4.db         1    181.000
## 6:   female_TRUE_5.db         1    184.567
## 7:   female_TRUE_7.db         1    231.133
## 8:   male_FALSE_25.db         1    184.634
```

Much more can be achieved using `data.table`, so I would strongly recommend to, at least, read the package introduction[3].

2.2 Loading one file

In this section, we will focus on loading data generated by pysolovideo[4] (i.e. `.db` files). In other words, we would like to load the data contained within a `.db` file as a data table structure such as the ones we described above. The function to load these data is called `loadPsvData`. It has many options, so it may be useful to read the documentation at some stage, but for now, we will just learn how to send a 'query' to obtain some data. I do not expect you to have already acquired any sort of data, so we will work with some of my samples, which I have enclosed within `rethomics`:

```
# let us unpack these couple of files
sample_files <- c("tube_monitor_validation_subset.db",
                  "monitor_validation_subset.db")
paths <- sapply(sample_files, loadSampleData)
# the path should be unpacked in a temporary directory:
print(paths)
```

```
##                              tube_monitor_validation_subset.db
## "/tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db"
##                                   monitor_validation_subset.db
##      "/tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db"
```

```
# The simplest case: we would like to load all the regions the first file:
dt <- loadPsvData(paths[1],verbose=F)
# We could also load all regions from all files:
dt <- loadPsvData(paths,verbose=F)
print(dt)
```

[2]tutorial available at `http://user2014.stat.ucla.edu/files/tutorial_Matt.pdf`
[3]`http://cran.r-project.org/web/packages/data.table/vignettes/datatable-intro.pdf`
[4]`https://github.com/gilestrolab/pySolo-Video`

```
##                                                                path region_id
##       1:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db         1
##       2:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db         1
##       3:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db         1
##       4:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db         1
##       5:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db         1
##      ---
## 2402338: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db        20
## 2402339: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db        20
## 2402340: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db        20
## 2402341: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db        20
## 2402342: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db        20
##                              experiment_id       t         x          y          w
##       1:      monitor_validation_subset.db     2.0 0.4914966 0.02380952 0.02380952
##       2:      monitor_validation_subset.db     2.5 0.5034014 0.02380952 0.03401361
##       3:      monitor_validation_subset.db     3.0 0.5476190 0.02551020 0.04931973
##       4:      monitor_validation_subset.db     3.5 0.5952381 0.02551020 0.04931973
##       5:      monitor_validation_subset.db     4.0 0.6003401 0.02891156 0.04761905
##      ---
## 2402338: tube_monitor_validation_subset.db 84224.5 0.7478705 0.04940375 0.03236797
## 2402339: tube_monitor_validation_subset.db 84225.0 0.7563884 0.05110733 0.02214651
## 2402340: tube_monitor_validation_subset.db 84225.5 0.7342419 0.04429302 0.04770017
## 2402341: tube_monitor_validation_subset.db 84226.0 0.7529813 0.04940375 0.04770017
## 2402342: tube_monitor_validation_subset.db 84226.5 0.7632027 0.04599659 0.04940375
##                   h phi mlog_L_x1000 xy_dist_log10x1000 xor_dist has_interacted
##       1: 0.02040816 146      -306000               -306     1000          FALSE
##       2: 0.02040816   0     -1858000              -1858      580          FALSE
##       3: 0.02380952   6     -1339000              -1339     1000          FALSE
##       4: 0.02551020   0     -1308000              -1308      997          FALSE
##       5: 0.02551020   0     -2114000              -2114      346          FALSE
##      ---
## 2402338: 0.02044293  10     -1753000              -1753      529          FALSE
## 2402339: 0.02044293  54     -1963000              -1963      420          FALSE
## 2402340: 0.02044293   3     -1601000              -1601      761          FALSE
## 2402341: 0.01873935   3     -1663000              -1663      547          FALSE
## 2402342: 0.01873935   8     -1953000              -1953      402          FALSE
```

In real life, each file will have a unique name containing the date and time as well as the name of the machine that generated it. Therefore, file names are used as a unique identifier of the experiement (`experiment_id`). Often, each file will correspond for a different condition (or combination of variables) that you would like to include in the resulting data table. In order to do that, we can pass a *query* to `loadPsvData`. A query is simply a table that has a column named `path`, and other columns for arbitrary conditions:

```
query <- data.table(path=paths,
                    treatment=c("control", "drug_A"))
print(query)

##                                                          path treatment
## 1: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control
## 2:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A
```

When working with a large number of files/conditions, it would be more common to load query from a spreadsheet or a CSV file. Once the query is prepared, we can simply load all the associated data:

```
dt <- loadPsvData(query,verbose=F)
print(dt)

##                              experiment_id region_id
##       1:      monitor_validation_subset.db         1
##       2:      monitor_validation_subset.db         1
##       3:      monitor_validation_subset.db         1
##       4:      monitor_validation_subset.db         1
##       5:      monitor_validation_subset.db         1
##      ---
## 2402338: tube_monitor_validation_subset.db        20
## 2402339: tube_monitor_validation_subset.db        20
## 2402340: tube_monitor_validation_subset.db        20
## 2402341: tube_monitor_validation_subset.db        20
## 2402342: tube_monitor_validation_subset.db        20
##                                                               path treatment       t
##       1:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A     2.0
##       2:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A     2.5
##       3:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A     3.0
##       4:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A     3.5
##       5:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A     4.0
##      ---
## 2402338: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control 84224.5
## 2402339: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control 84225.0
## 2402340: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control 84225.5
## 2402341: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control 84226.0
## 2402342: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control 84226.5
##                  x          y          w          h phi mlog_L_x1000 xy_dist_log10x1000
##       1: 0.4914966 0.02380952 0.02380952 0.02040816 146      -306000               -306
##       2: 0.5034014 0.02380952 0.03401361 0.02040816   0     -1858000              -1858
##       3: 0.5476190 0.02551020 0.04931973 0.02380952   6     -1339000              -1339
##       4: 0.5952381 0.02551020 0.04931973 0.02551020   0     -1308000              -1308
##       5: 0.6003401 0.02891156 0.04761905 0.02551020   0     -2114000              -2114
##      ---
## 2402338: 0.7478705 0.04940375 0.03236797 0.02044293  10     -1753000              -1753
## 2402339: 0.7563884 0.05110733 0.02214651 0.02044293  54     -1963000              -1963
## 2402340: 0.7342419 0.04429302 0.04770017 0.02044293   3     -1601000              -1601
## 2402341: 0.7529813 0.04940375 0.04770017 0.01873935   3     -1663000              -1663
## 2402342: 0.7632027 0.04599659 0.04940375 0.01873935   8     -1953000              -1953
##          xor_dist has_interacted
##       1:     1000          FALSE
##       2:      580          FALSE
##       3:     1000          FALSE
##       4:      997          FALSE
##       5:      346          FALSE
##      ---
## 2402338:      529          FALSE
## 2402339:      420          FALSE
## 2402340:      761          FALSE
## 2402341:      547          FALSE
## 2402342:      402          FALSE
```

Importantly, there is now a column named ‘treatment’ in your data. This means you can then conveniently study differences explained by that variable. You could imagine a scenario where you have done 50 experiments where you have changed both sex and genotype, and want to investigate the effect of these variables on a behavioural variable (e.g. amount of sleep).

2.3 Fancier queries

In some cases, you may want to load only specific regions, and that may have different conditions, within the same experiment. If you add a column in your query called `region_id`, only specified regions will be loaded (as opposed to all regions in the previous example).

```
query <- data.table(path=paths,
                    treatment=c("control", "drug_A"),
                    region_id=rep(1:20,each= 2))

# We make a dummy query where we imagine that
# every other region contains a female individual
```

```
query[, sex := ifelse(region_id %% 2, "male", "female" )]

##                                                            path treatment region_id    sex
##  1: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control         1   male
##  2:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A         1   male
##  3: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control         2 female
##  4:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A         2 female
##  5: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control         3   male
##  6:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A         3   male
##  7: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control         4 female
##  8:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A         4 female
##  9: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control         5   male
## 10:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A         5   male
## 11: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control         6 female
## 12:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A         6 female
## 13: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control         7   male
## 14:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A         7   male
## 15: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control         8 female
## 16:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A         8 female
## 17: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control         9   male
## 18:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A         9   male
## 19: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control        10 female
## 20:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A        10 female
## 21: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control        11   male
## 22:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A        11   male
## 23: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control        12 female
## 24:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A        12 female
## 25: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control        13   male
## 26:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A        13   male
## 27: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control        14 female
## 28:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A        14 female
## 29: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control        15   male
## 30:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A        15   male
## 31: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control        16 female
## 32:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A        16 female
## 33: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control        17   male
## 34:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A        17   male
## 35: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control        18 female
## 36:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A        18 female
## 37: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control        19   male
## 38:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A        19   male
## 39: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control        20 female
## 40:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A        20 female
##                                                            path treatment region_id    sex

# Note that we load only regions lower than twenty
print(query)

##                                                            path treatment region_id    sex
##  1: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control         1   male
##  2:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A         1   male
##  3: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control         2 female
##  4:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A         2 female
##  5: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control         3   male
##  6:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A         3   male
##  7: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control         4 female
##  8:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A         4 female
##  9: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control         5   male
## 10:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A         5   male
## 11: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control         6 female
## 12:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A         6 female
## 13: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control         7   male
## 14:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A         7   male
## 15: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control         8 female
## 16:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A         8 female
## 17: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control         9   male
## 18:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A         9   male
## 19: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control        10 female
## 20:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A        10 female
## 21: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control        11   male
## 22:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A        11   male
## 23: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control        12 female
## 24:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A        12 female
## 25: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control        13   male
## 26:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A        13   male
## 27: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control        14 female
## 28:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A        14 female
## 29: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control        15   male
## 30:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A        15   male
## 31: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control        16 female
## 32:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A        16 female
## 33: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control        17   male
## 34:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A        17   male
## 35: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control        18 female
## 36:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A        18 female
## 37: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control        19   male
## 38:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A        19   male
## 39: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control        20 female
## 40:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A        20 female
##                                                           path treatment region_id    sex

# Now we can use our query
dt <- loadPsvData(query,verbose=F)
print(dt)

##                                                                path treatment region_id
##       1:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A         1
##       2:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A         1
##       3:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A         1
##       4:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A         1
##       5:      /tmp/RtmpF9uSlD/db_files/monitor_validation_subset.db    drug_A         1
##      ---
## 2095923: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control        20
## 2095924: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control        20
## 2095925: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control        20
## 2095926: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control        20
## 2095927: /tmp/RtmpF9uSlD/db_files/tube_monitor_validation_subset.db   control        20
##             sex                    experiment_id       t         x          y          w
##       1:   male      monitor_validation_subset.db     2.0 0.4914966 0.02380952 0.02380952
##       2:   male      monitor_validation_subset.db     2.5 0.5034014 0.02380952 0.03401361
##       3:   male      monitor_validation_subset.db     3.0 0.5476190 0.02551020 0.04931973
##       4:   male      monitor_validation_subset.db     3.5 0.5952381 0.02551020 0.04931973
##       5:   male      monitor_validation_subset.db     4.0 0.6003401 0.02891156 0.04761905
##      ---
## 2095923: female tube_monitor_validation_subset.db 84224.5 0.7478705 0.04940375 0.03236797
## 2095924: female tube_monitor_validation_subset.db 84225.0 0.7563884 0.05110733 0.02214651
## 2095925: female tube_monitor_validation_subset.db 84225.5 0.7342419 0.04429302 0.04770017
## 2095926: female tube_monitor_validation_subset.db 84226.0 0.7529813 0.04940375 0.04770017
## 2095927: female tube_monitor_validation_subset.db 84226.5 0.7632027 0.04599659 0.04940375
##                   h phi mlog_L_x1000 xy_dist_log10x1000 xor_dist has_interacted
##       1: 0.02040816 146      -306000               -306     1000          FALSE
##       2: 0.02040816   0     -1858000              -1858      580          FALSE
##       3: 0.02380952   6     -1339000              -1339     1000          FALSE
##       4: 0.02551020   0     -1308000              -1308      997          FALSE
##       5: 0.02551020   0     -2114000              -2114      346          FALSE
##      ---
## 2095923: 0.02044293  10     -1753000              -1753      529          FALSE
## 2095924: 0.02044293  54     -1963000              -1963      420          FALSE
## 2095925: 0.02044293   3     -1601000              -1601      761          FALSE
## 2095926: 0.01873935   3     -1663000              -1663      547          FALSE
## 2095927: 0.01873935   8     -1953000              -1953      402          FALSE
```

Note that there were some warnings, this is because some of the requested regions are not in the example data. Indeed, I only provided a small subset of available regions in order to reduce the size of the samples. Also, as before, our data table(`dt`) has additional columns for the conditions we added.

As a heads-up, lets see how one would use data table syntax to compute median x position per sex and treatment:

```
summary <- dt[, .(median_x = median(x)),
                by=c("sex","treatment")]
print(summary)

##       sex treatment  median_x
## 1:   male    drug_A 0.3867121
## 2: female    drug_A 0.2670068
## 3:   male   control 0.3959044
## 4: female   control 0.4726962
```

2.4 Loading from network drive

Now, this is great when you know where you `.db` files are, but in real life, they may be stored by `pysolovideo` in a network drive, and you cannot retrieve their locations easily. What you do know, is when (i.e. at what date) you started your experiment, and which device you have used. Using the function `fetchPsvResultFiles`, you can retrieve data from the directory structure. If you want to list/retrieve all available experiments/files, you can do:

```
# You will need to change this
YOUR_RESULT_DIR <- '/data/psv_results'
all_files <- fetchPsvResultFiles(YOUR_RESULT_DIR)
print(all_files[,.(file,date,machine_name)])

##                                                          file                date
##   1: 1970-01-01_00-01-03_00036dfce6e94dee9bb1a845281b086e.db 1970-01-01 00:01:03
##   2: 2015-04-08_14-36-49_00036dfce6e94dee9bb1a845281b086e.db 2015-04-08 14:36:49
##   3: 2015-04-08_15-45-55_00036dfce6e94dee9bb1a845281b086e.db 2015-04-08 15:45:55
##   4: 2015-04-09_15-13-44_00036dfce6e94dee9bb1a845281b086e.db 2015-04-09 15:13:44
##   5: 2015-04-13_15-58-19_00036dfce6e94dee9bb1a845281b086e.db 2015-04-13 15:58:19
##  ---
## 115: 2015-06-05_17-19-00_00036dfce6e94dee9bb1a845281b086e.db 2015-06-05 17:19:00
## 116: 2015-06-05_17-20-55_00036dfce6e94dee9bb1a845281b086e.db 2015-06-05 17:20:55
## 117: 2015-06-13_14-05-59_00046dfce6e94dee9bb1a845281b086e.db 2015-06-13 14:05:59
## 118: 2015-06-13_14-06-10_00016dfce6e94dee9bb1a845281b086e.db 2015-06-13 14:06:10
## 119: 2015-06-13_14-07-44_00036dfce6e94dee9bb1a845281b086e.db 2015-06-13 14:07:44
##      machine_name
##   1:     GGSM-003
##   2:     GGSM-003
##   3:     GGSM-003
##   4:     GGSM-003
##   5:     GGSM-003
##  ---
## 115:     GGSM-003
## 116:     GGSM-003
## 117:     GGSM-004
## 118:     GGSM-001
## 119:     GGSM-003
```

`YOUR_RESULT_DIR` is where ever your `psv_results` directory is on your computer, or on the network. You can also generate a query(table) in which you request experiments by `date` and a `machine_name`. It is crucial that the query has columns exactly named `date` and `machine_name`:

```r
# we want two experiments performed the same day (2015-06-02),
# in two different machines "GGSM-001" and "GGSM-003":

query <- data.table(date="2015-06-05",
                    machine_name=c("GGSM-001","GGSM-003", "GGSM-004"),
                    condition=c("A", "B","C"))
# Note that I added an extra column to map an experimental condition:
print(query)

map <- fetchPsvResultFiles(YOUR_RESULT_DIR, query)
# This should be able to find the requested files, if they exist.
print(map)

# Importantly, the added condition is still there,
# so we can simply send this map to loadPsvData:
dt <- loadPsvData(map)
```

It is possible that you have performed several experiments the same day. If the date is ambiguous, *the latest experiment* will be returned (and a warning is displayed). You can also specify the date as '2015-06-02_hh-mm-ss', which should be unambiguous.

2.5 Analysing data

2.6 Sleep annotation

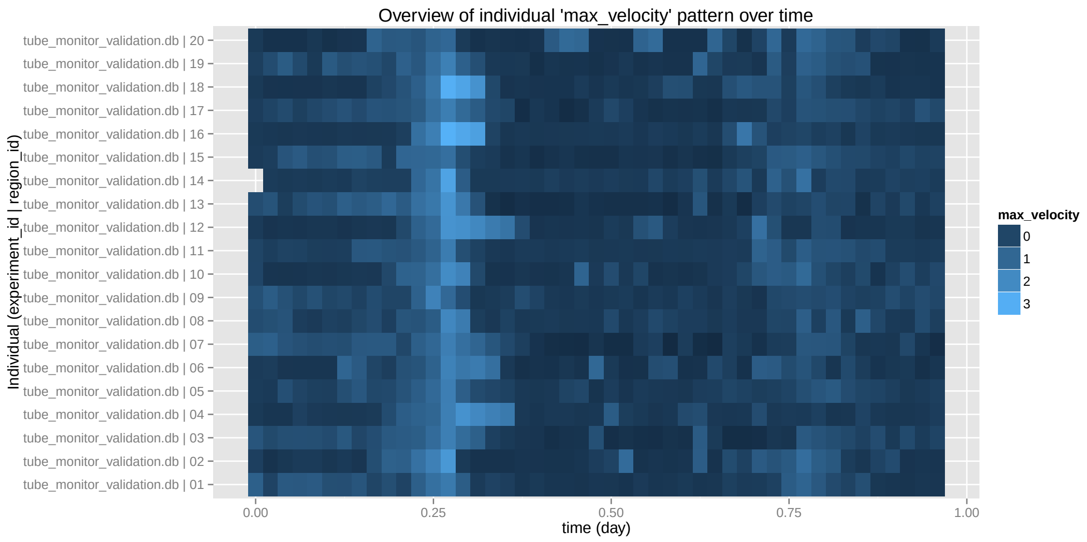
2.7 Visualisation

This is the exciting part! Now we know how to load data, I will use preloaded data in the package. The `tube_monitor_validation` data set is one experiement with 20 regions.

2.7.1 Overview plot

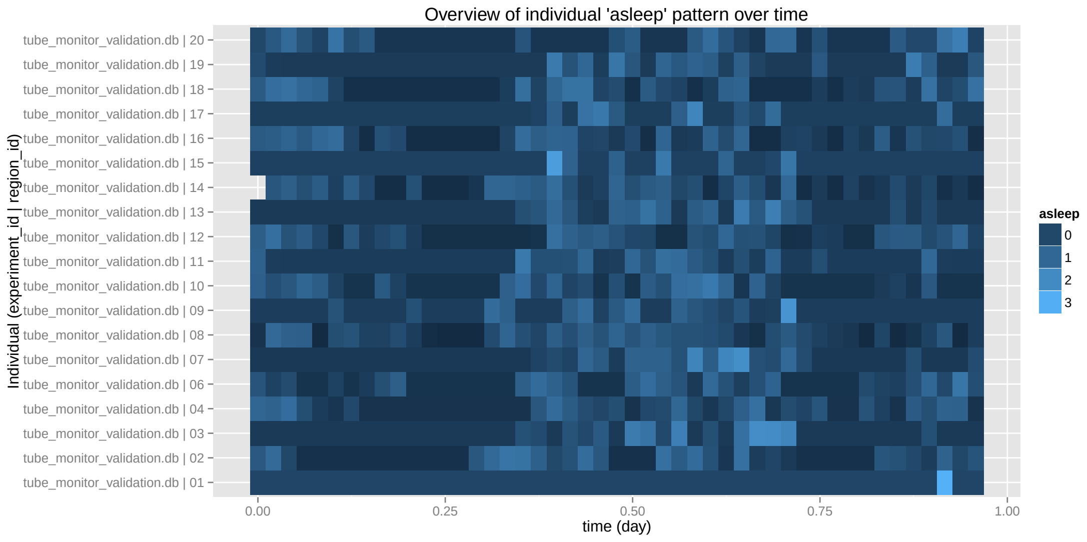
We would like to have an overview of the activity in each region as a graphical quality control.

```r
data(tube_monitor_validation)
# We make a sleep annotation, for ach animal.
# This will also compute maximal velocity for every chunk of 10s
my_data <- tube_monitor_validation[,sleepAnnotation(.SD),
                                   by=key(tube_monitor_validation)]
pl <- overviewPlot(max_velocity,my_data)
print(pl)
```



The colour intensity is proportional to the value of `max_velocity`, but you may like to represent another variable:

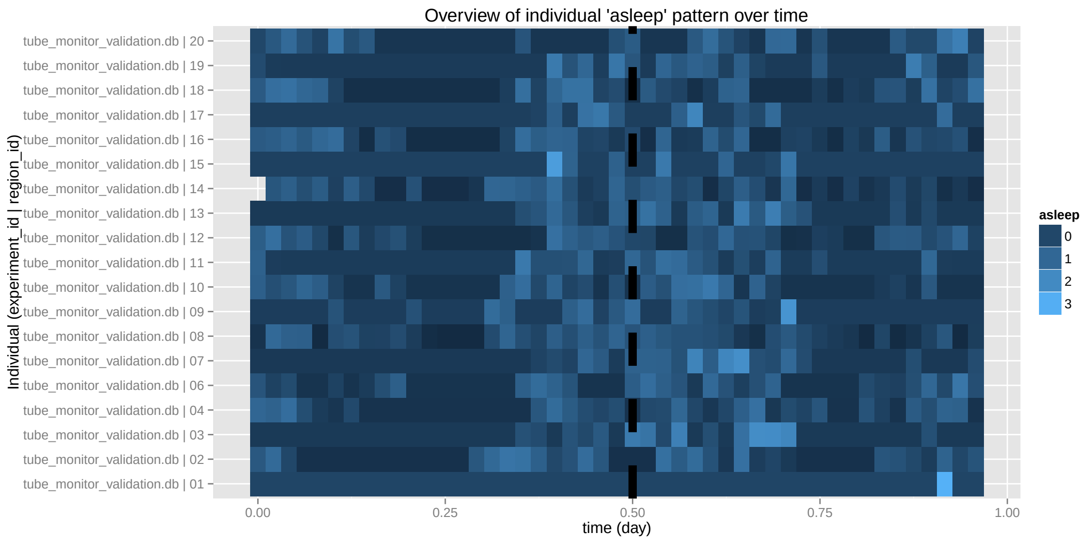
```
data(tube_monitor_validation)
pl <- overviewPlot(asleep,my_data)
print(pl)
```



All built in visualisation functions decribed here will produce a ggplot[5] object, which you can easyly modify. for instance, drawing a doted line at 12h (0.5 days).

```
pl <- pl + geom_vline(aes(xintercept=c(0.5)),linetype=2,size=3)
print(pl)
```

[5] `http://ggplot2.org/book/`



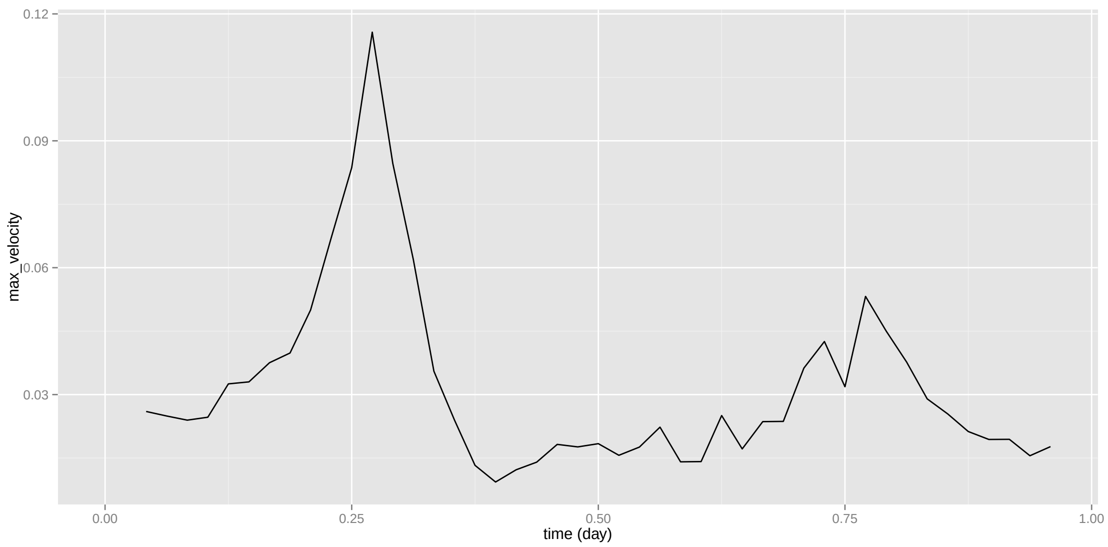
Many other options such as grouping are available for `overviewPlot`.

```
?overviewPlot()
```

2.7.2 Ethograms

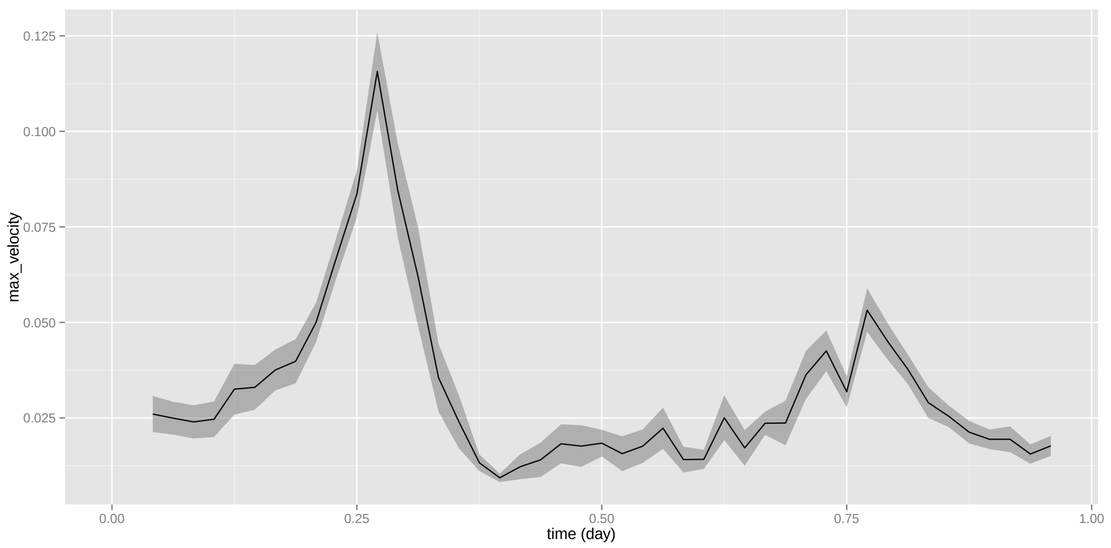
Most of the time you will want to see difference between population overtime. Ethograms will represent a chosen y variable over time. In addition, it will group data by time window(the default is 30min):

```
pl <- ethogramPlot(max_velocity, my_data)
print(pl)
```



Let us add some error bar:

```
pl <- ethogramPlot(max_velocity,my_data,error_bar="sem")
print(pl)
```



0.125
0.100
0.075
0.050
0.025
max_velocity
0.00
0.25
0.50
0.75
1.00
time (day)