1 Running the simulation

1.1 Load up `Speckles.jl`

```
using Speckles
```

1.2 Provide dictionary with parameters

- Each parameter must be an array

```
νHα2 = [456810,456813] #GHz
paramDict = Dict(
                :n    => collect(10:10:100),# number of atoms
                :νm   => [νHα2], # line frequencies in GHz
                :Em   => ["ones"], # relative line magnitudes
                :σ    => [20.0], # Doppler broadening in GHz
                :fγ   => [2.0e6],#,"shot10%","shot50%",10.0,1.0,0.16], # mean photon count rate in GHz
                :deadtime   => [0.0], # detector deadtime in nanoseconds
                :resolution => [0.010],#,0.10], # detector resolution in nanoseconds
                :jitter     => [0.015], # detector timing jitter in nanoseconds
                :efficiency => [0.9], # detector efficiency
                :darkcounts => [1.0e-8], # detector dark count rate in GHz
                :duration   => [20.0], # duration of each correlation measurement in nanoseconds
                :window     => ["halfwindow"], # time over which to average correlations in nanoseconds
                :repeat     => [100], # number of times to repeat correlation measurement
                :reinstance => [true] # control whether or not frequencies and phases should be reinstanced between measurements
                )
```

```
Dict{Symbol, Vector{T} where T} with 14 entries:
  :n          => [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
  :jitter     => [0.015]
  :reinstance => Bool[1]
  :duration   => [20.0]
  :fγ         => [2.0e6]
  :efficiency => [0.9]
  :σ          => [20.0]
  :window     => ["halfwindow"]
  :darkcounts => [1.0e-8]
  :resolution => [0.01]
  :νm         => [[456810, 456813]]
  :Em         => ["ones"]
  :deadtime   => [0.0]
  :repeat     => [100]
```

1.3 Run the simulation and access the dictionary

- Takes all combinations of parameters possible with given arrays and runs each as a simulation

- Appends the results of the simulation in the simulation database

- Each series of simulations gets its own id
- Each simulation gets its own id
- Each simulation, its parameters, and its results gets its own row in the database
- Plots and data are stored in a directory with the same name as the simulation
- Returned table contains parameters from the current series of simulations
- All simulation can be retrieved with `load_db()`

```
tbl = Speckles.run(paramDict)
```

```
Error: MethodError: no method matching Speckles.SpeckleSim(::DateTime, ::Ba
se.UUID, ::Speckles.SpeckleParams{Float64}, ::Speckles.Beamsplitter, ::Floa
t64, ::Vector{Speckles.SpeckleReadout}, ::Vector{Speckles.Correlation})
```

1.4 Select data from the table

- Get the latest run

```
# sort the table by the number of atoms for plotting reasons
tbl_nsort = sort(tbl,:n)

# get column with number of atoms
nvals = select(tbl_nsort,:n)

# get snr columns

cols_str = string.(colnames(tbl))
singlecols = occursin.("single",cols_str)
singlecols = [singlecols...]
single_colnames_str = cols_str[singlecols]
single_snr_cols = Symbol.(single_colnames_str)
plt = plot()
for col in single_snr_cols
    plot!(plt,nvals,select(tbl_nsort,col),label=string(col))
end
plot!(plt,title="SNR for single correlation simulation")
```

```
Error: UndefVarError: tbl not defined
```

Let's save that bad boy too!

```
savefig(plt,"badboy.svg")
```

```
Error: UndefVarError: plt not defined
```