Supplementary Information

Grajales-Reyes et al.

SI Materials and Methods

Voluntary wheel running. The onset and progression of weakness in both animal models (αC418W and WT) was monitored to determine the effects of statin treatment. A computer-monitored mouse activity wheel system (wheel counter model 86061, wheel diameter 12.7 cm, clear polycarbonate cage, USB computer interface model 86056A, activity wheel monitor software version 9.2, Lafayette Instruments, Lafayette) was used to determine exercise and locomotor activity profile of mice during treatment. This system monitored the average velocity of the activity wheel during 24 hours. The computer logged the average velocity (meters/minute) and the cumulative distance (meters) the mouse traveled for every second along the course of these 24 hours. Once this file was obtained, the file was opened in Excel and the entire A column was filtered to display the average velocity, since this was the variable that was going to be analyzed. After this, all the data contained in the file, except the average velocity, was erased so that only a column containing all the average velocities could be saved as a coma separated value (CSV) file. This file was then renamed from a .csv to a .dat file so that it could be analyzed in a custom made program. This program analyzes the moments in which the activity wheel's velocity was greater than 0 and calculates two values. First it calculates for how long (seconds) the mouse ran and second it calculates the average velocity of this activity period (meters/minute). Once these measurements were performed, the activity period duration and its corresponding average velocity were logged in side-by-side columns. This was organized so that the activity period duration is in an ascending order starting with the lowest value, which is always 1 second. Activity periods that exhibit the same duration were all displayed with their respective average velocity. For detailed information about the custom program refer to the below description. This code was written in the C++ programming language and can be compiled in the Bloodshed Dev-C++ software to produce an executable (.exe) file that is the custom program *per se*. These values were then analyzed with Sigma Plot (Systat Software Inc., San Jose, CA), which divides a scatter plot generated with the aforementioned (activity interval duration and its respective average velocity) data into a 10 x 10 grid. Then, Sigma Plot calculated the frequency of the data points contained within each grid unit. The obtained frequency values were then used in MatLab 7.4 R2007a (The MathWorks, Inc., El Segundo, CA) to produce contour plots. Velocity bin values were on the X-axis and time bin values (in log scale) are on the Y-axis, and the frequency information was displayed as colored contours. Increased frequencies were represented as a shift from blue to red contours. In order to display the dynamic range of the data, 15 contours were distributed following a cubic curve with a final contour level that displays a maximum frequency of 22. As such, the final contour (22) contains all data equal or greater than its threshold value. This arrangement allows for low frequency contours to be closer spaced than high frequency contours, providing more detail in the areas of the histogram representing mouse activity. Before starting the experiment mice were placed in a similar cage with a similar activity wheel so that they learned how to run prior to the first experiment day. Once the activity recording finished (24 hours) the mouse was returned to its original cage. In order to prepare the cage for a new animal, each cage and activity wheel was washed with tap water and cleaned with 70 % alcohol after each experiment day; the bedding was also changed.

Animals, care and procedures. Male 6-8 weeks-old mice that express the αC418W mutation on the muscle nAChR, and WT (FVB) were used. FVB mice were used as controls for αC418W and αV249F while C57BL/6 was used for δS262T since these mutant mice were created using these respective background strains. All the stable transgenic mice have been inbred for vastly more than 15 generations, inheriting the transgene in a simple Mendelian fashion. Mutant transgenic mice were previously established and described in detail[43]. All animals were bred and housed in an environmentally controlled facility (10/14-h light/dark cycle, temperature between 20-22°C, relative humidity 65-75%) and have free access to food (Harlan Laboratories, IN) and tap water. All protocols were approved by the University of Puerto Rico Institutional Animal Care and Use Committee (IACUC). To screen the transgenic mice, genomic DNA was recovered from mouse tail tips using DNeasy kit (Qiagen) following manufacturer instructions. The presence of the transgene was determined by polymerase chain reaction (PCR) using PCR beads (GE) and primers to amplify α and δ subunit genes and the NEO gene. The NEO gene primer was used to identify every transgenic line since only these mice have the gene, and the second primer varied upon which subunit contained the mutation[43]. PCR products were visualized in agarose gel electrophoresis. Upon completion of experiments, all animals were euthanized by cervical dislocation and disposed according to institutional policies.

Statin treatment. Freshly prepared Atorvastatin calcium (Lipitor®) (44 mg/kg) (5 mg/ml) or placebo (PBS, 1X) was administered intragastrically via oral gavage with a metal feeding tube (Popper & Sons, Inc., NY) daily up to 36 days.

Electromyography. Evoked compound muscle action potential (CMAP) responses were recorded in mice weighting 20-30 g using a Dual Bio Amp/Stimulator coupled to a Power Lab 4/30 data acquisition system (ADInstruments, CO) under Avertin anesthesia as described by Gomez et al., 1997[28]. The CMAP responses were generated by the sciatic nerve stimulation. In order to do this an incision lateral and parallel to the femur was performed. This incision exposed the sciatic nerve, to which a copper wire was encircled. After this, the copper wire was coupled to an electrode that delivered a train pulse of 10 stimuli at a frequency of 5 Hz during of 0.05 ms. The percentage of decrease in amplitude (mV) of the CMAP (decrement) was calculated using the amplitude (peak positive to peak negative) of the 1st and 10th responses.

Confocal microscopy imaging. For the NMJ size measurement, images were collected in the Confocal Imaging Facility at the University of Puerto Rico (CIF-UPR) using a Zeiss LSM 510 Laser Scanning Confocal Microscope (Carl Zeiss, Inc.). Endplates were labeled by incubating in Alexa-Fluor® 488-conjugated α-bungarotoxin (Invitrogen) for 1 hour and washed 3 times with PBS1X (15 min). Motor endplates were visualized using a 40X objective. Zeiss LSM 510 parameters were optimized at the beginning of every tissue sample observation. In order to obtain a good representation of the endplate population, the hemidiaphragms were divided into 5 sections, dividing the space between the ventral and dorsal part of the hemidiaphragm equally. Once all the images were obtained, 10 Z-stacks were acquired per mouse. Each one of these sections was imaged with the aforementioned parameters. Collected Z-stacks were analyzed using the Imaris x64 6.1.3 software (Bitplane Inc., CT) in which a surface was generated over the reconstructed endplates so that its size could be calculated in three-dimensions. These measurements were then plotted as normalized histograms so that changes in the sample distribution could be observed. These histograms were fitted using Peakfit (Systat Software Inc., CA). In order to perform the caveolin-1 (Cav-1) staining the *tibialis anterior* muscle was used. Once dissected, it was rapidly dipped in 2-methyl-butane (Sigma-Aldrich) bathed by liquid nitrogen. Once frozen, tissues were mounted in OCT compound so that the muscle could be cut in 10 μm slices using a cryostat (Leica, model CM1100, Leica, IL). Then, tissues were fan dried for 20 minutes and immersed into an acetone-methanol (1:1) mixture for 20 minutes at –20 °C. Following fixation, tissues were fan dried once again for 20 minutes. In order to block the tissues, muscle slices were immersed in blocking solution (2% NGS, 0.2% Triton X-100, 1% DMSO in PBS 1X) for 1 hour. To prepare the slides for the antibody addition a circular area was drawn around the tissue slice with a PAP pen, which creates a thin-filmed hydrophobic barrier that keeps the antibody solution localized. Once the hydrophobic film dried, the antibody solution (caveolin-1 antibody H-97, Santa Cruz Biotechnologies, diluted 1:500 in blocking solution) was added for 12-16 hours at 4°C. In order to wash the primary antibody, the tissue was immersed in washing buffer (0.05% Tween-20 in PBS 1X) 3 times for 10 minutes each. Finally, the secondary antibody (Molecular Probes, goat anti rabbit 1:1000) was added for one hour at 25°C, and as before the tissue was washed 3 times 10 minutes each. Later, mounting medium for fluorescence with DAPI (H-1200, Vector Laboratories Inc.) was added. All Cav-1 imaging was performed in a TCS laser-scanning microscope (Leica, IL). The percentages of Cav-1 positive endplates were measured in order to compare the effects of the statin treatment and the difference on Cav-1 positive endplates between WT and αC418W mice.

Glyoxal-bis (2-hydroxyanil) stain (GBHA). GBHA histochemical staining was performed according to Kashiwa et al. 1964[49] and Gomez et al., 2002[29]. In brief, the *tibialis anterior* muscle was frozen in 2-methyl-butane bathed by liquid nitrogen followed by mounting in OCT compound and sliced in a cryostat at a 10 μm thickness, stained and mounted. Each slice was stained in the following order: slice #1 (cholinesterase stain), slice #2 (GBHA, calcium stain), slice #3 (cholinesterase stain). The slice (1 or 3) that exhibits the highest endplate number was selected and compared against slice #2 (GBHA-stained). Cholinesterase was stained by immersing the slices in a modified Ringer's solution (0.1% $CuSO_4 \cdot 5H_2O$, 0.2% glycine and 5 mM acetylcholine iodide adjusted to pH 6.5 with a few drops of a 10% solution of 2-amino-2-metylpropa-1-ol). After 15 minutes in Ringer's solution at room temperature the slices were rinsed in distilled water and placed in a 1% solution of yellow ammonium sulfide (pH 9) for 5 seconds, followed by distilled water rinse and subsequent immersion in ethanol 70%. The calcium stain was prepared by mixing 16 ml 0.4% glycoxal bis-2-hydroxyanil dissolved in methanol with 7.2 ml NaOH 5%. Then, slides were immersed in this solution and air dried for 2 minutes followed by additional immersion and air dry for two minutes to finally rinse in 70% ethanol. After this, the slides were dipped in 0.25% methylene blue dissolved in 70% ethanol. The counterstained sections were dehydrated in acetone, cleared in xylene, and mounted.

Caspase-3 activity experiments. A firefly luciferase-based assay was used to measure activity of caspase 3 (Caspase-Glo® 3/7, Promega). Muscles were homogenized (25 mM HEPES pH 7.5, 0.1% (v/v) Triton X-100, 5 mM $MgCl_2$, 2 mM 1,4-dithiothreitol, 10 mM NH_4Cl, 10 mM 3-methyladenine, 74 μM antipain, 0.15 μM aprotinin, 1.3 mM EDTA, 20 μM leupeptin, and 15 μM pepstatin). After homogenization, a 20 μg protein product was added to the luminometer in triplicates for the protease luminescence assay.

Cholesterol measurement in muscle. 800 μl of each sucrose gradient fraction were subjected to the Bligh-Dyer method for the extraction of lipids in solution[50]. Briefly, 3.75 ml 1:2 (v/v) $CHCl_3$: MeOH were added to each sample, followed by 1.25 ml of $CHCl_3$, and 1.25 ml of distilled water; after each addition, samples were vigorously vortexed. The organic phase of each sample was carefully extracted and dried under $N_{2(g)}$. Cholesterol was separated from other lipids on rhodamine 6G stained silica gel G plates with petroleum ether/diethyl ether (98:2, v/v) as the solvent system. The spots corresponding to cholesterol was extracted with petroleum ether:ethyl ether (2:3 v/v) and further assayed using the Wako cholesterol E Kit (Wako Chemicals USA, VA) according to the manufacturer's indications.

Statistical Analysis. All experiments were replicated at least three times, with the number of replicates (n) indicated in the figure legend. Each replicate represents a mouse, and each data point was the average of at least three different samples. Bars in all figures represent the standard error of the mean (SEM). T-tests were performed using GraphPad Prism version 4.00 for Windows, GraphPad Software, San Diego, California, USA.

Custom software code

```
/*********************************************************************
//                                                          *
// Programmer: Wilfredo F. García Beltrán                        *
// Laboratory of José A. Lasalde Dominicci
// File: Time-Velocity Tracing Analysis                       *
//                                                          *
//*********************************************************************
/*
    This program converts a list of continuous second-to-second
velocities into a list of time intervals (delimited by sub-threshold
velocities) with their respective average velocities. It accepts as
input a tab-delimited list of continuous time-points (in seconds) with
an instanteneous velocity for every time point; the data must be in a
*.dat file. The output is also a *.dat file.
*/

#include <iostream>
using namespace std;
#include<fstream>
#include <iomanip>
#include <cmath>

struct meanvelocity
{
   int timeinterval ;
   double meanvelocity ;
} ;

// Prototypes of functions and subfunctions
double Ask_for_threshold(bool&) ;
```

```
void Time_interval_mean_velocities(void) ;
  void Read_and_group_data_with_mean_velocity(double) ;
  void Read_groups_with_mean_velocity(ifstream&, meanvelocity [], int) ;
  int Count_elements_with_mean_velocity(ifstream&) ;
  void Display_array_and_create_file_with_mean_velocity(meanvelocity [], int, ofstream&) ;
  void Order_with_mean_velocity(meanvelocity [], int) ;
int Count_elements(ifstream&) ;
void Display_array(int [], int ) ;
void Finishing_message (void) ;

int main ()
{
   system ("cls") ;
      cout << "\n\t<<<  Analysis of Time-Velocity Tracings  >>>\n"
          << "\n       This program converts a list of continuous second-"
          << "\n  to-second velocities into a list of time intervals (delimited"
          << "\n  by sub-threshold velocities) with their respective average"
          << "\n  velocities. It accepts as input a tab-delimited list of"
          << "\n  continuous time-points (in seconds) with an instanteneous"
          << "\n  velocity for every time point; the data must be in a *.dat"
          << "\n  file. The output is also a *.dat file.\n\n" ;
   system ("pause") ;

   Time_interval_mean_velocities() ;

   Finishing_message() ;

   return 0 ;
}

// Definition of functions

void Finishing_message (void)
{
    system ("cls") ;
    cout << "\n  ***Thank you for using this program***\n" << endl ;
    system ("pause") ;
}

double Ask_for_threshold(bool& check)
{
   char selection ;
   double threshold ;
   do
```

```
    {
        system ("cls") ;
        cout << "\n  Do you wish to set a threshold value to determine which values"
             << "\n  of instantaneous velocity are significant (default = 0) (Y/N)---> ";
        cin >> selection ;
    } while ((selection != 'Y') && (selection != 'N')) ;
    if (selection == 'Y')
    {
        cout << "\n\n  Specify the 'threshold' value (non-negative real number)---> " ;
        cin >> threshold ;
        if(cin.fail()IIthreshold < 0)
        {
            check = true ;
            cout << "\n\nInvalid entry. This program will close."
                 << " Try again...\n" << endl ;
            system("pause") ;
        }
    }
    if (selection == 'N')
    {
        threshold = 0 ;
    }

    return threshold ;
}

void Time_interval_mean_velocities(void)
{
    system("cls") ;
    //Ask for threshold value
    double threshold ;
    bool check = false ;
    threshold = Ask_for_threshold(check) ;
    if(check == true)
    {
        exit(1) ;
    }

    // Read and group data from an archive and then send the resulting
    // groups to an another external intermediary archive
    Read_and_group_data_with_mean_velocity(threshold) ;

    // Read from the intermediary archive created to then make a data array
    ifstream entrada ;
```

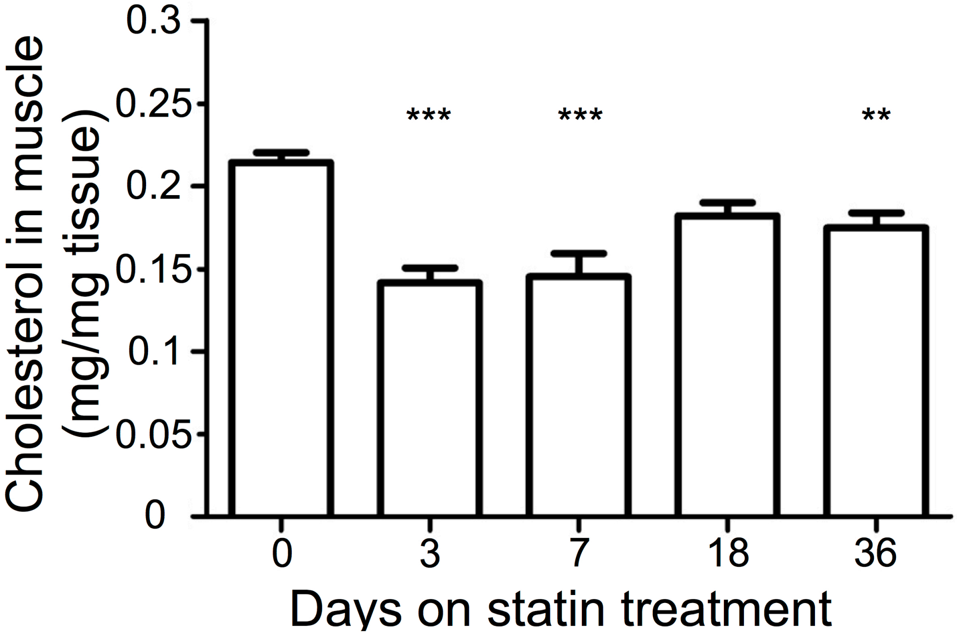
```
   ofstream salida ;
   entrada.open("intermediary.dat") ;

  // Ask for the name for the archive where the analyzed data will be sent
   string filename ;
   cout << "\n  Indicate the filename you prefer for the file that will be"
       << "\n  created for the data to be sent to (.dat) ---> " ;
   cin >> filename ;
   salida.open(filename.c_str()) ;

   // Determine the amount of elements (groups)
   int number_of_elements ;
   number_of_elements = Count_elements_with_mean_velocity(entrada) ;
   cout << "\n  There are " << number_of_elements
       << " groups (time intervals)" << endl << endl ;
   system("pause") ;

   entrada.close() ;

   // Read the elements (groups) and create an array
   ifstream entrada2 ;
   entrada2.open("intermediary.dat") ;
   meanvelocity ordered[number_of_elements] ;
   Read_groups_with_mean_velocity(entrada2, ordered, number_of_elements) ;

   // Order the elements in the array
   Order_with_mean_velocity(ordered, number_of_elements) ;

   Display_array_and_create_file_with_mean_velocity(ordered, number_of_elements, salida) ;

   system("pause") ;
}

void Read_and_group_data_with_mean_velocity(double n)
{
   ifstream entra ;            // Intermediate file
   string filename ;           // Input file name
   ofstream sali ;             // Output file "intermediary.dat"

   do
   {   entra.clear() ;
      cout << "\n  Indicate the filename from which the data will"
          << " be extracted (.dat) ---> " ;
      cin >> filename ;
```

```
        // Abrir el archivo
        entra.open(filename.c_str()) ;
        if(!entra)
        {
            cout << "  The file named " << filename << " is not"
                << " available or doesn't exist. Try again.\n" ;
            system("pause") ;
        }
    } while(entra.fail()) ;

    sali.open("intermediary.dat") ;

    int contador = 0 ;
    double numero ;
    double mean = 0 ;

    entra >> numero ;
    mean = numero ;
    while(!entra.eof())
    {
        if(numero <= n)
        {
            mean = 0 ;
            contador = 0 ;
            entra >> numero ;
        }
        else
        {
            mean = (numero+(mean*contador))/(contador+1) ;
            contador++ ;
            entra >> numero ;
            if(numero <= n ll entra.eof())
            {
                sali << contador << "\t" << mean << endl ;
            }
        }
    }
    entra.close() ;
    sali.close() ;
}

int Count_elements_with_mean_velocity(ifstream& entrada)
{
    int quantity = 0 ;
```

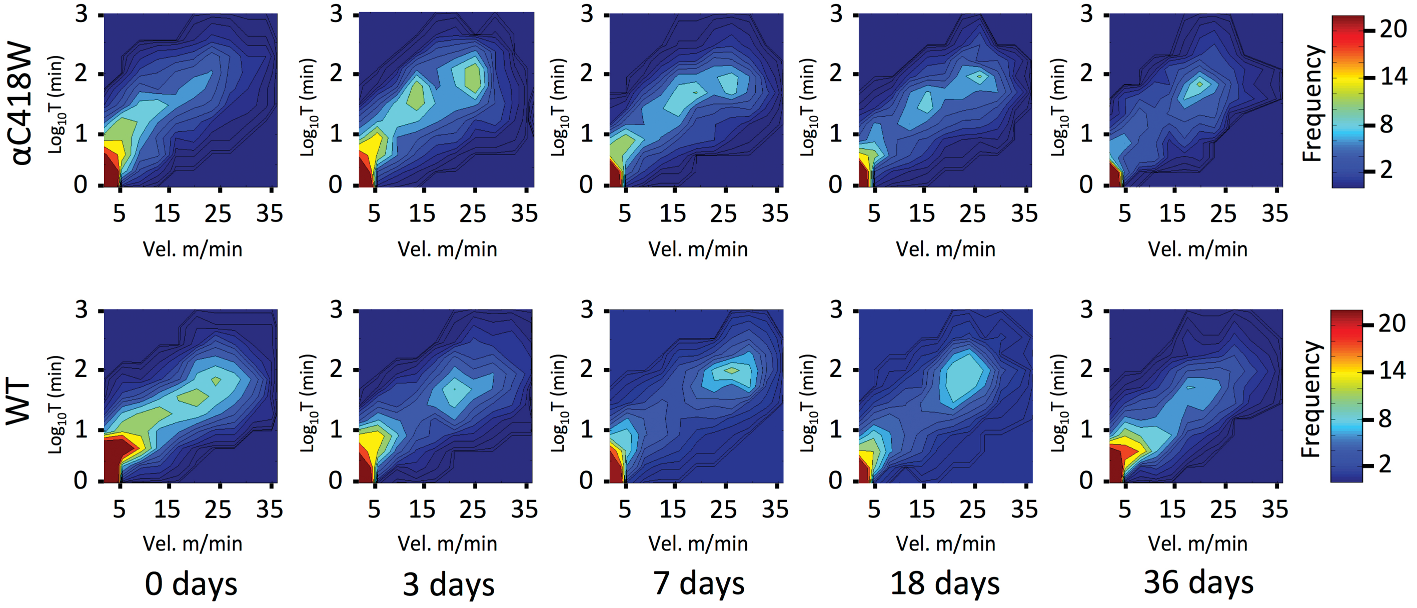
```
   double element ;

   entrada >> element >> element ;

   // Count the number of elements
   while(!entrada.eof())
   {
      quantity++ ;
      entrada >> element >> element ;
   }
   return quantity ;
}

void Read_groups_with_mean_velocity(ifstream& entrada, meanvelocity orden[], int cant_e)
{
   for(int i = 0 ; i < cant_e ; i++)
   {
      entrada >> orden[i].timeinterval >> orden[i].meanvelocity ;
   }
}

void Display_array_and_create_file_with_mean_velocity(meanvelocity orden[], int cant_e,
ofstream& sali)
{
   sali << "Time interval\tMean velocity" << endl ;
   for(int i = 0 ; i < cant_e ; i++)
   {
      sali << orden[i].timeinterval << "\t\t\t\t" << orden[i].meanvelocity << endl ;
      cout << "Time interval: " << orden[i].timeinterval
          << "\tMean velocity: " << orden[i].meanvelocity << endl ;
   }
}

void Order_with_mean_velocity(meanvelocity ordered [], int n)
{
   // Cycle to organize the data
   for (int pase = 1 ; pase < n ; pase++)
   {
       for (int compa = 0 ; compa < (n-pase) ; compa++)
       {
         if (ordered[compa].timeinterval > ordered[compa+1].timeinterval)
         {
           swap (ordered[compa].timeinterval, ordered[compa+1].timeinterval) ;
           swap (ordered[compa].meanvelocity, ordered[compa+1].meanvelocity) ;
```

```
            }
        }
    }
}
```

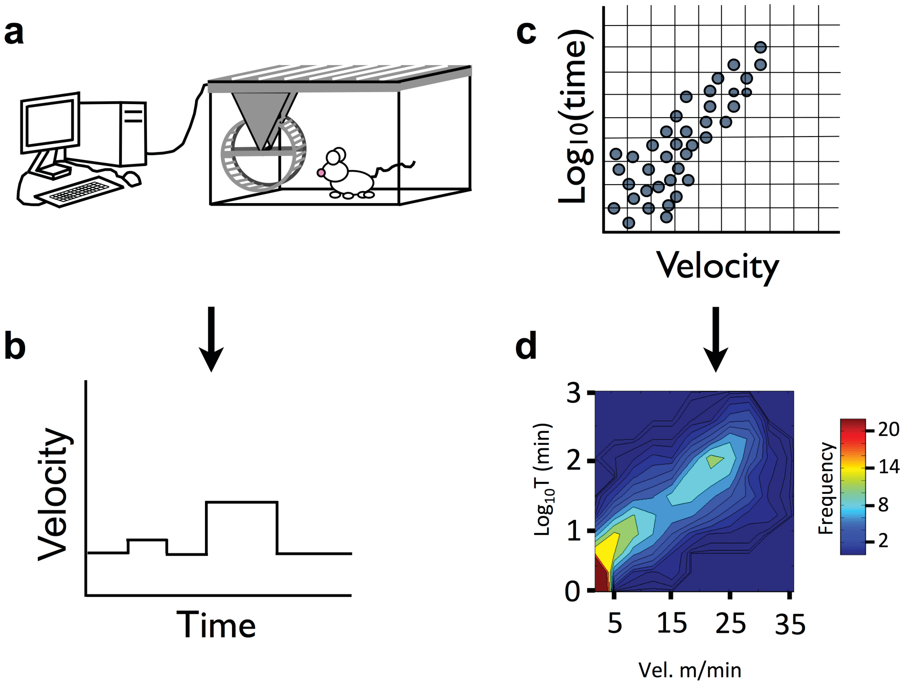
Supplementary Figures



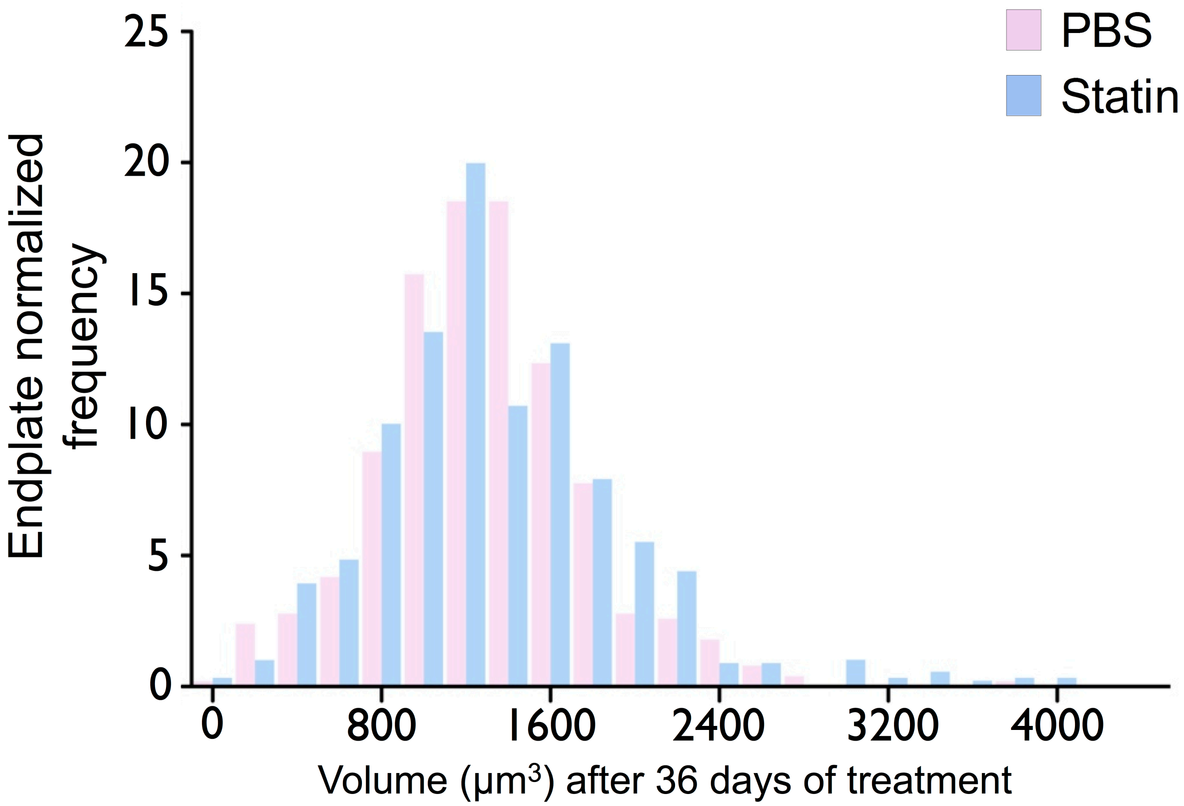
Supplementary Figure S1 Statin treatment significantly decreases muscle cholesterol content. Organic extractions from the *tibialis anterior* muscle show a significant cholesterol content reduction (***$P < 0.001$, **$P < 0.01$) (n = 20, 4, 4, 5, 9 for 0, 3, 7, 18 and 36 days of statin treatment respectively).



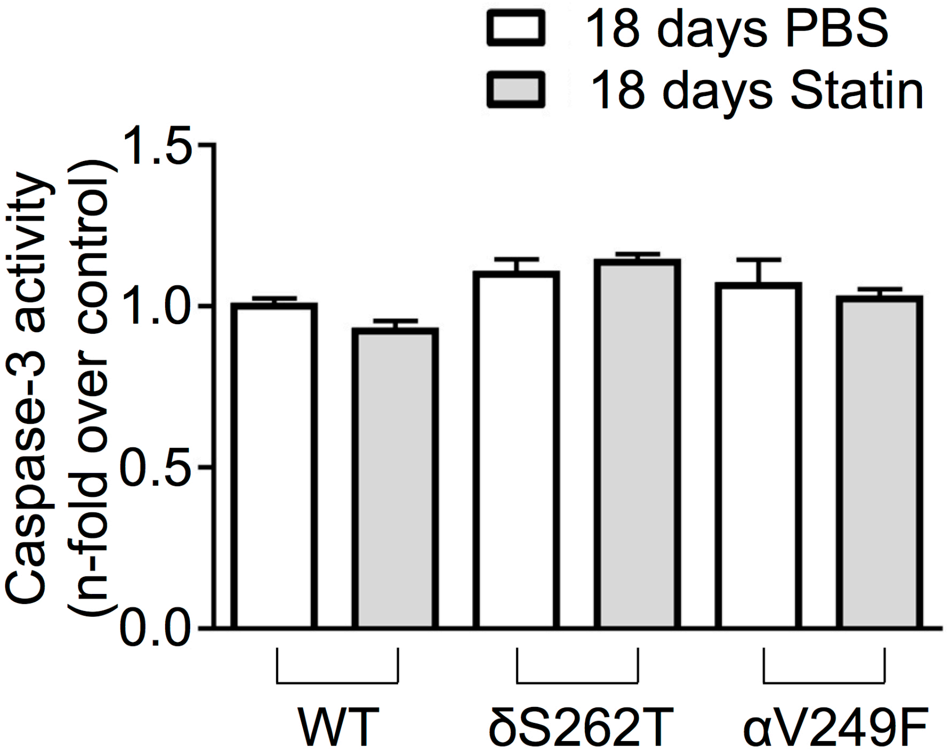
Supplementary Figure S2 PBS treated mice display a normal locomotor activity even after 36 days of treatment. The locomotor activity was assessed by means of detailed measurement of time spent running (non-stop) in the activity wheel, and the corresponding velocity in a 24-hour period. WT and αC418W mice were treated with PBS daily and on days 0, 3, 7, 18 and 36 were placed in cages with activity wheels (n = 4, 4, 9, 8, 4 for WT and n = 4, 4, 4, 4, 4 for αC418W in 0, 3, 7, 18 and 36 days of treatment, respectively). Time running and corresponding velocities were recorded and plotted in heat maps. The Y-axis corresponds to Log_{10}time spent running (non-stop) and the X-axis is the corresponding velocity. Color corresponds to number of occasions the mice ran for a particular period of time and velocity.



Supplementary Figure S3 Cartoon representation of the locomotor activity analysis. **(a)** This cartoon represents the cage with the activity wheel in which the mice run voluntarily for a 24-hour period. (**b**) Once the data was acquired from the computer, it was analyzed in a custom made program that generates the data that shows the average velocity during periods of constant activity. (**c**) After this, the data was used to produce a scatter plot. Then, the Sigma Plot software divided the scatter plot chart area into a 10 x 10 grid and measured the frequency of every division of the grid. (**d**) This data was finally analyzed in Matlab to construct the heat maps.



Supplementary Figure S4 Statin treated αC418W and PBS treated WT endplate size distributions are similar. The endplate size distribution of the αC418W statin treated mice, although different form the PBS-treated αC418W mice, becomes similar to the WT PBS-treated mice after 36 days of statin treatment (n= 3 for WT PBS and αC418W statin treated mice).



Supplementary Figure S5 WT**,** αV249F and δS262T muscle caspase-3 activity is not affected by the statin treatment. In order to test whether the caspase-3 activity in other SCS mutations such as the αV249F and δS262T increase in response to cholesterol depletion, the corresponding levels were measured at the initial time point in which the αC418W caspase-3 levels showed sensitivity to the statin treatment. Like WT mice, none of the other SCS mice showed a response to the treatment (n = 3 in WT, αV249F and δS262T mice in PBS and statin treatment).