UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

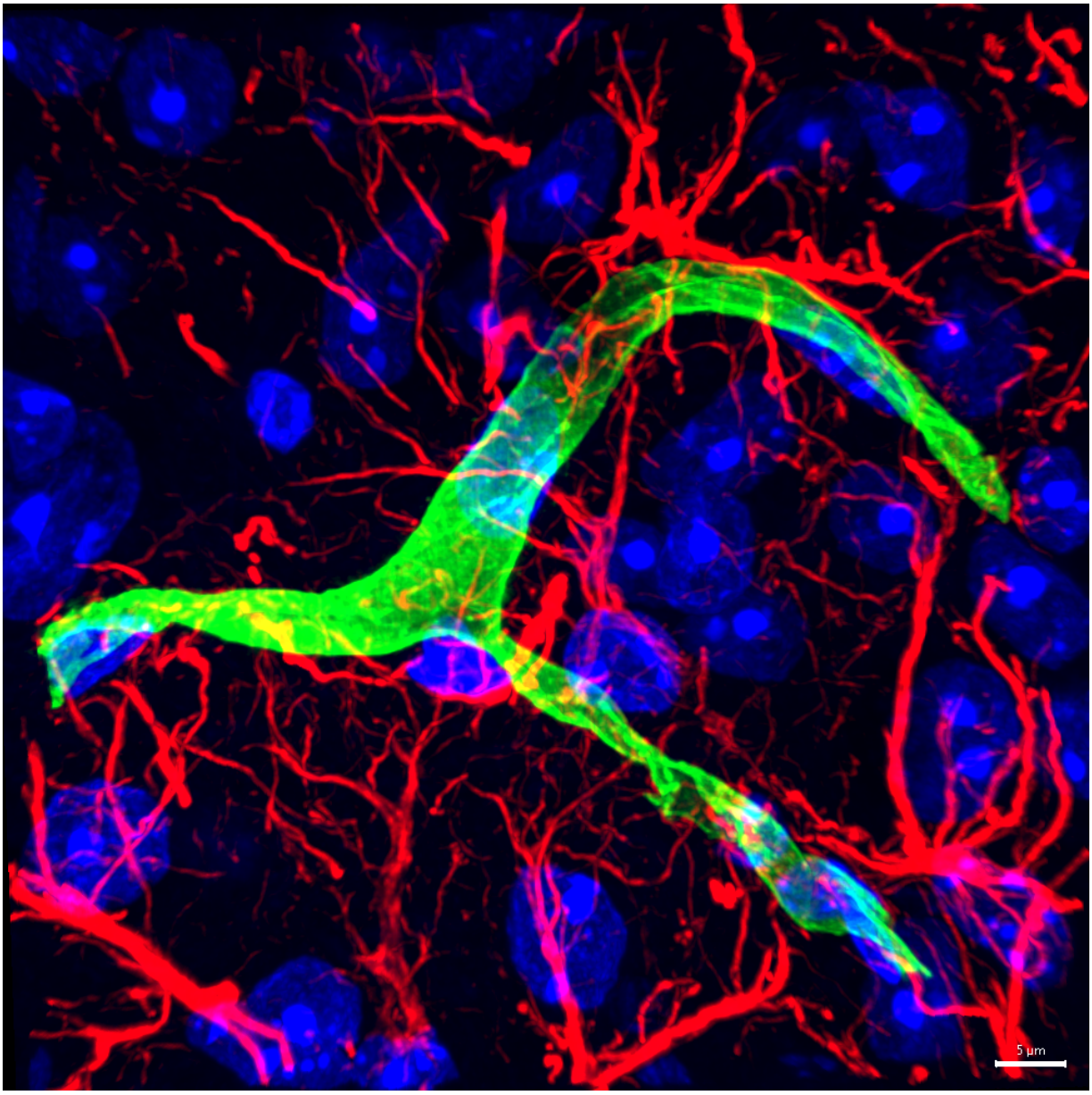
SBC.00121 Biochemistry Laboratory Rotation

David Parker

On the polarization of aquaporin-4 during the circadian cycle

Supervised by Prof. Albrecht and Katrin Wendrich

September 2, 2021



Abstract

In this study we recreated an experiment published by Hablitz et al., 2020, which analysed the circadian control over the perivascular polarization of the water channel protein AQP-4. In our experiment, we were able to replicate the results but also approached the data analysis in a different manner which lead us to question the method used in Hablitz et al., publication. We also found, using mice with a abnormal circadian clock (total Per2 knockouts), that these mutants may have a more active glymphatic system than wild type mice. This finding may be a good starting point for further studies involving the glymphatic system and its relationship with the circadian rhythm.

Contents

1 General introduction

In 2013, an article was published by Maiken Nedergaard and colleagues describing for the first time the importance of sleep in the clearance of metabolic waste from the brain. The studied pathways form a waste clearance system in the brain, which was named the *glymphatic system*, due to its homologous function to the lymphatic system and dependance on glial cells. It was found to be more active during the rest phase. The system functions via astrocytic aquaporin-4 (AQP-4) water channels.[20] In 2020, an other study involving the glymphatic system was lead by Nedergaard. This time the circadian control of brain glymphatic and lymphatic fluid flow was studied, by analysing the perivascular polarization of AQP-4 at different stages of the circadian cycle in mice. The results left them with the conclusion that the distribution of cerebrospinal fluid (CSF) in mouse brains is indeed controlled by the circadian rhythm and that the rhythm is supported by AQP-4.[11]

In the 2020 paper, experiments performed on mice submitted to constant light for 10 days supported the hypothesis that the glymphatic rhythm is controlled by the circadian cycle and not just the light-dark cycle. The authors stated and provided data that the constant light does not have an enormous effect on the animals circadian behavior.[11] For this project, the experiments analysing the perivascular polarization of AQP-4 were recreated with wild type and with Per2 knockout mice to test the influence of the circadian cycle on the process. Per2 is a core clock gene involved in the regulation of the circadian cycle in mammals, making the mutant mice suitable test subjects for the experiment. The hypothesis is that the KO mice will not show the same rhythm in the perivascular polarization of the AQP-4 in the glymphatic system.

A total of 16 mice were analysed, using immunohistochemistry (IHC) and laser scanning confocal microscopy (LSCM). It was found that the circadian rhythm does effect the perivascular polarization of AQP-4 in the glymphatic system of mice.

1.1 Circadian Rhythm

Circadian rhythms, or circadian cycles, are endogenous biological processes that oscillate periodically within many organisms. These rhythms govern behaviours such as **sleep**, metabolism and other vital systems.[8][5] In most species, the circadian cycle has a period of 24h. The 24h cycle has evolved to match the light/dark cycle created by the earth rotating around its axis, otherwise known as day and night. The day/night cycle has a strong effect on the environment and the organisms within, such as plants only photosynthesizing in sunlight, or some species being evolved to be well adapted to nocturnal hunting. Having an internal clock allows the organism to anticipate and physiologically prepare for the coming activity which gives it an advantage over other organisms lacking such a system. This has been shown with the growth of cyanobacteria.[18]

Mammalian circadian rhythms are controlled by many circadian clocks that together form the circadian system. The circadian system has a hierarchical structure, with the brain and liver being the main sources of synchronization for the rest of the cellular clocks. One of the strongest external stimuli is light captured by the eyes which in turn stimulates the suprachiasmatic nuclei (SCN) in the brain. The SCN is the master clock which synchronises the other clocks in the brain.[3]

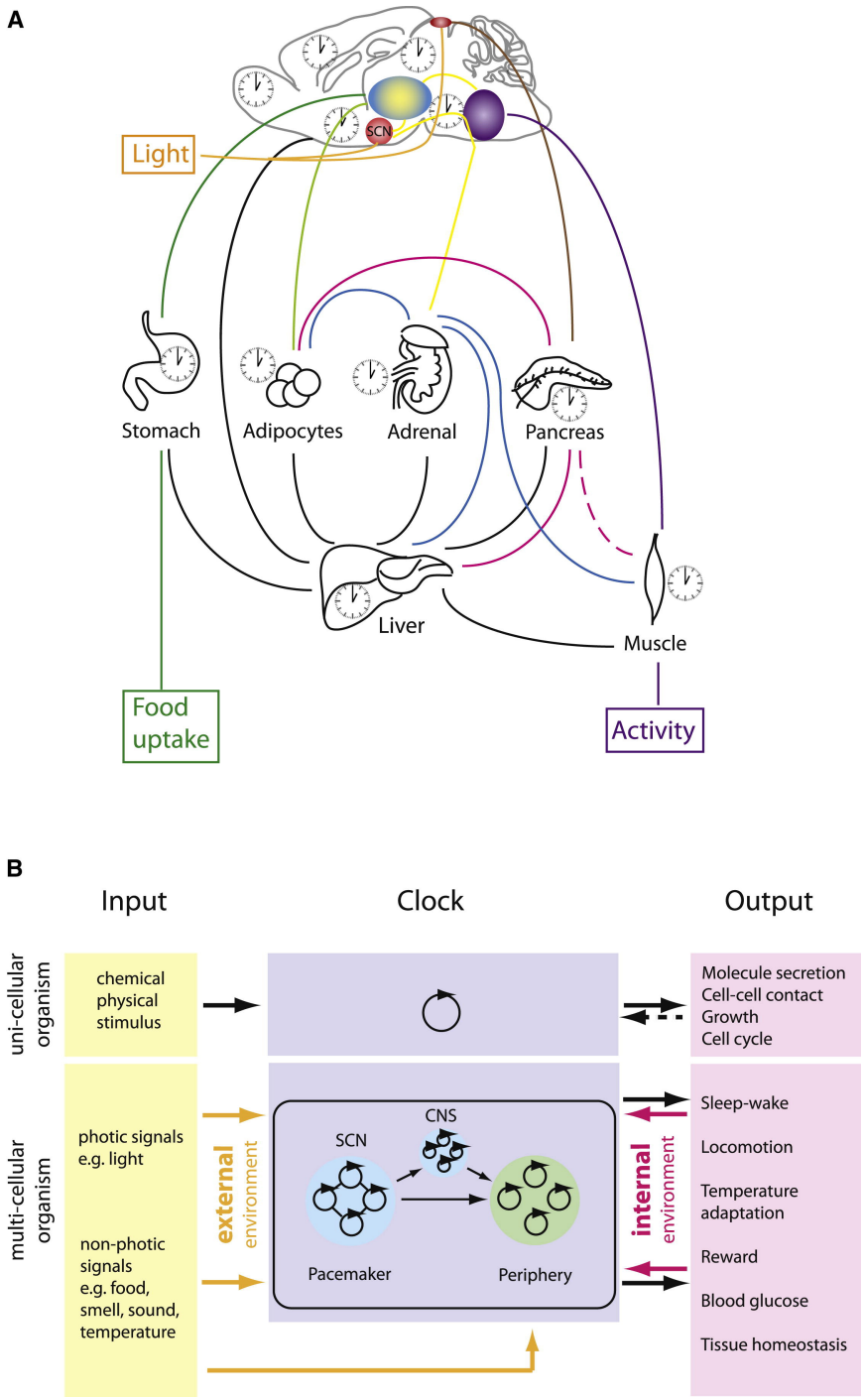


Figure 1: Organization of the CircadianSystem
(A) The master clock in the SCN (red circle) receives input from light stimulus (orange line) which then synchronises the central clocks in the brain. The different clocks and regions themselves regulate different functions ((yellow-blue shaded oval : metabolic and reward integration, purple oval : motor coordination).
(B) Subdivision of the circadian system : input to the clock, clock mechanism, and clock output, at cellular (top) and systemic (bottom) level. Adapted from Albrecht, 2012 [3]

Period 2 (Per2) is an important gene in the molecular regulation of circadian clocks at cellular level. The gene allows for the central clock to be reset in response to light stimulus.[21] Mice having had the Per2 gene rendered completely disfunctional (TPer2KO) express shorter circadian periods, and if the mice are kept in constant darkness (DD - dark dark), the circadian rhythmicity is lost.[22]

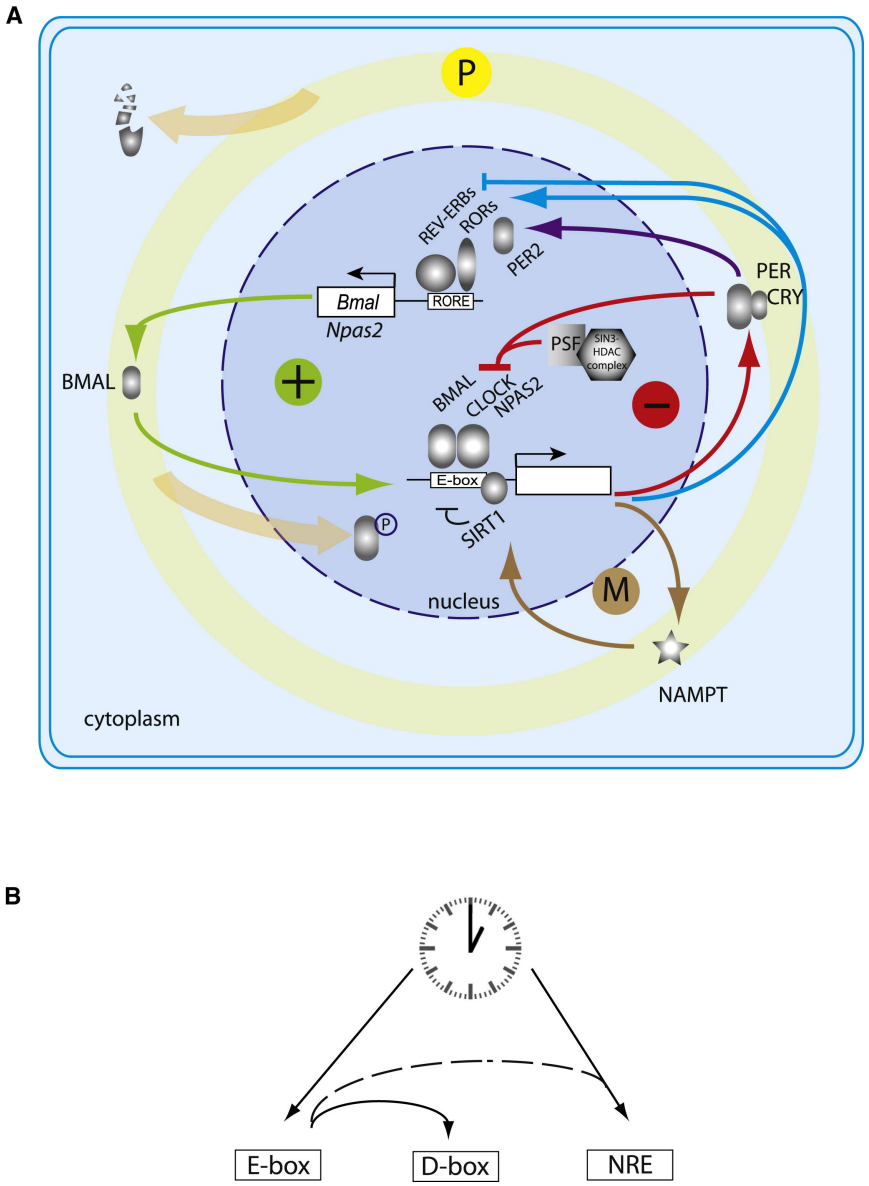


Figure 2: Molecular Circadian Clock Mechanisms in a Cell
(A) The clock mechanism consists of two main parts: (1) a transcriptional-translational feedback loop (TTL) consisting of a positive (green +) and a negative (red -) limb and (2) oscillating posttranslational modification of gene products in the TTL (yellow circle marked with P), which regulate degradation and/or nuclear localization of these proteins (orange arrows).
The positive and negative limbs are intertwined via clock protein-driven nuclear receptors (blue lines) and their interactions with PER2, a component of the negative limb (purple arrow). A metabolic oscillator (brown M) is driven by the TTL and feeds back on it via SIRT1.
(B) Promoter elements in clock controlled genes (CCGs). The CCGs are regulated either directly or indirectly. (1) Direct regulation via BMAL/CLOCK binding at E-boxes or REV-ERB/ROR binding at RORE-elements and (2) Indirect regulation via binding of clock-regulated PAR-ZIP factors (e.g., DBP) on D-elements, or via protein-protein interactions between PER2 and nuclear receptors (hatched line) at nuclear receptor elements (NREs) such as ROREs. Taken from Albrecht, 2012[3]

1.2 Glymphatic System

The glymphatic system uses perivascular tunnels, composed of astroglial cells surrounding capillaries, to continuously promote the interchange of cerebrospinal fluid (CSF) and Interstitial fluid (ISF) (ISF) which facilitates the evacuation of metabolic waste and soluble proteins from the central nervous system (CNS). This system also allows the distribution of essential molecules throughout the brain, such as glucose, lipids, amino acids and hormones.[16] The interchange of CSF and ISF is mainly powered by the water channel proteins aquaporin-4 (AQP-4), which are localised to the astrocytic endfeet surrounding the brain vasculature.[14]

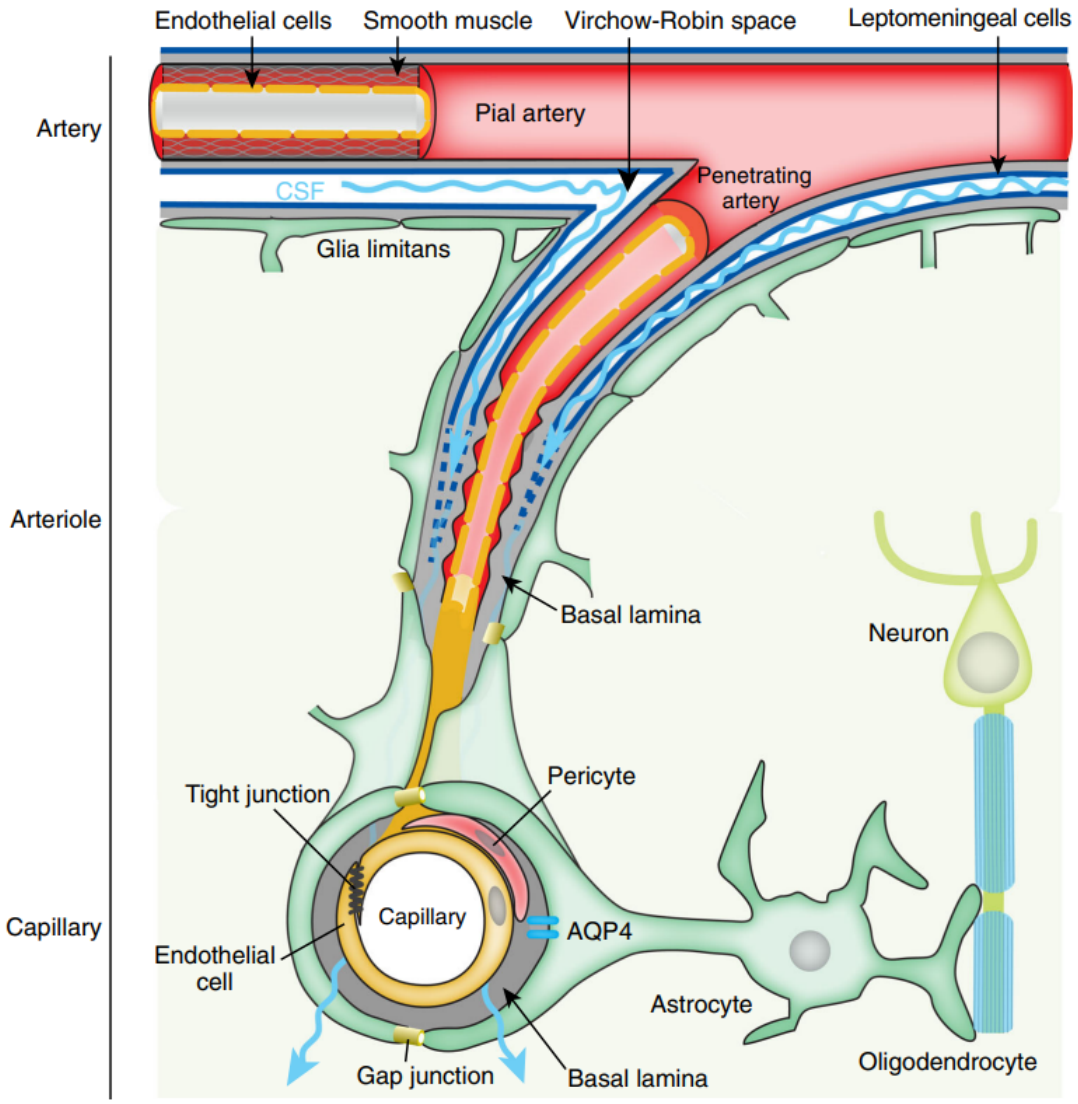


Figure 3: The structure and function of the neurovascular unit allow bidirectional communication between the microvasculature and neurons, with astrocytes playing intermediary roles. Astrocytic vascular endfeet expressing aquaporin-4 (AQP-4) surround the entire vasculature and form the boundary of the perivascular spaces. Adapted from Jessen et al., 2015[16]

1.3 Aquaporins

Aquaporins are transmembrane proteins which selectively allow water to diffuse through them. They form one of the 3 subfamilies of the water channel protein (WCP) family. The remaining two subfamilies being aquaglyceroporins which also are permeable to small uncharged molecules

such as glycerol, and S-aquaporins which do not contain the highly conserved asparagine-proline-alanine motif which is known as the 'signature' sequence of WCPs. The water channel protein family is part of the membrane intrinsic protein superfamily.[4]

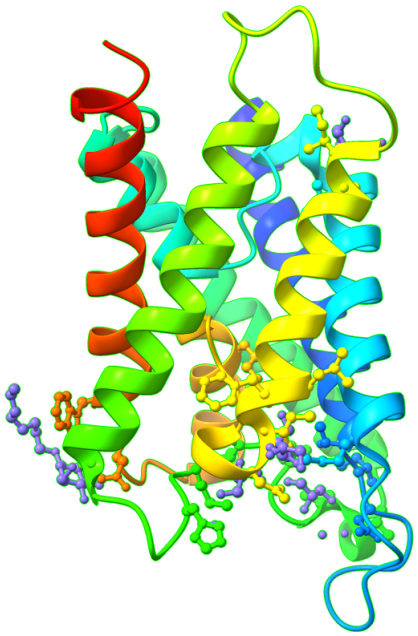


Figure 4: Ribbon representation of AQP-4. Adapted from Ho et al., 2009[12]

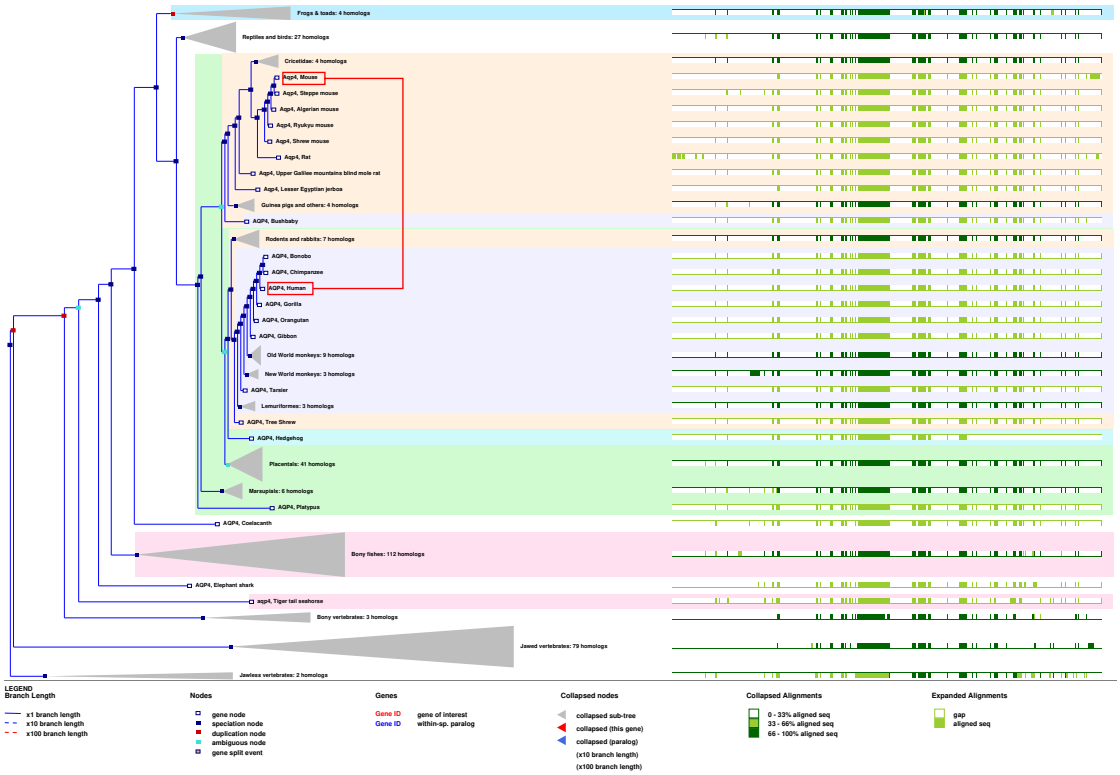


Figure 5: Ensembl gene tree showing the phylogenic evolutionary history of AQP-4. Modified to highlight the human and mouse relationship from the Ensembl database.[13]

AQP-4 is the most abundant water channel in the brains of mammals. The proteins are localised to the astrocytic endfeet and processes, which are in contact with the capillaries of

the blood-brain barrier and the synapses respectively.[12]

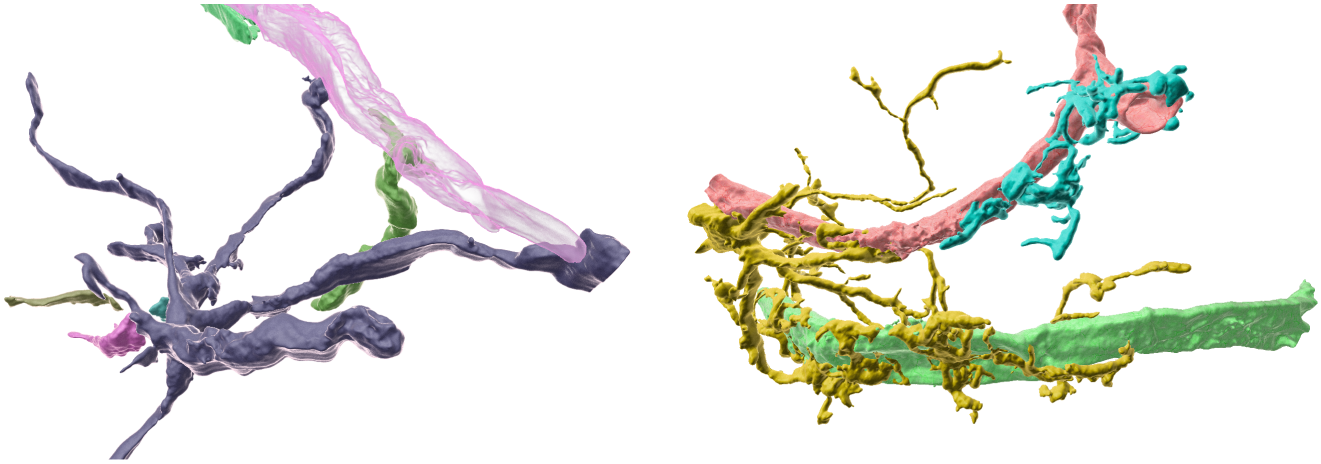


Figure 6: 3D reconstruction of astrocytes and blood vessels from a mouse brain immunohistochemically stained (DAPI, AQP-4, GFAP and wheat germ agglutinin (WGA)) and imaged by laser scanning confocal microscopy (LSCM). The blood vessels are coloured in pink (left) and blue and red (right) and are surrounded by several astrocytes coloured in blues and greens (left) and yellow and blue (right). The part of the astrocytes which makes contact with the blood vessel are known as 'astrocytic endfeet'

2 Goal

The goal of this project is to compare the amount of AQP-4 protein localised to the blood vessels in mice brains under two different conditions :

- Active phase (ZT18) versus rest phase (ZT6).
- Control mice (Per2flfl) versus mutant mice (TPer2KO).

Where ZT stands for Zeitgeber, 'time giver'. From ZT0 to ZT12, the animals are in a light environment and from ZT12 to ZT18 in a dark environment. Here the light, or absence of, serves as the external cue to which the mice synchronize their circadian clocks.

Thus the effect of the time of day and the genotype is begin compared and if there is an interaction between the two.
We hypothesize that the activity of the water transport channels is higher during the rest phase of circadian cycle, for mice this corresponds to the light phase. This is being tested by staining the AQP-4 water channels in the mice brains with a fluorescent marker, and then quantify the fluorescent intensity relative to the background fluorescence with LSCM. Wild type mice are expected to show a higher relative AQP-4 staining intensity at the astrocytic endfeet at time point ZT6 I_{ZT6} (middle of the light phase) in comparison to ZT18 I_{ZT18} (middle of the dark phase) :

$$I_{ZT6} > I_{ZT18} \tag{2.1}$$

If this is not the case, then there should be no observable difference between the two time points :

$$I_{ZT6} \leq I_{ZT18} \tag{2.2}$$

3 Material and methods

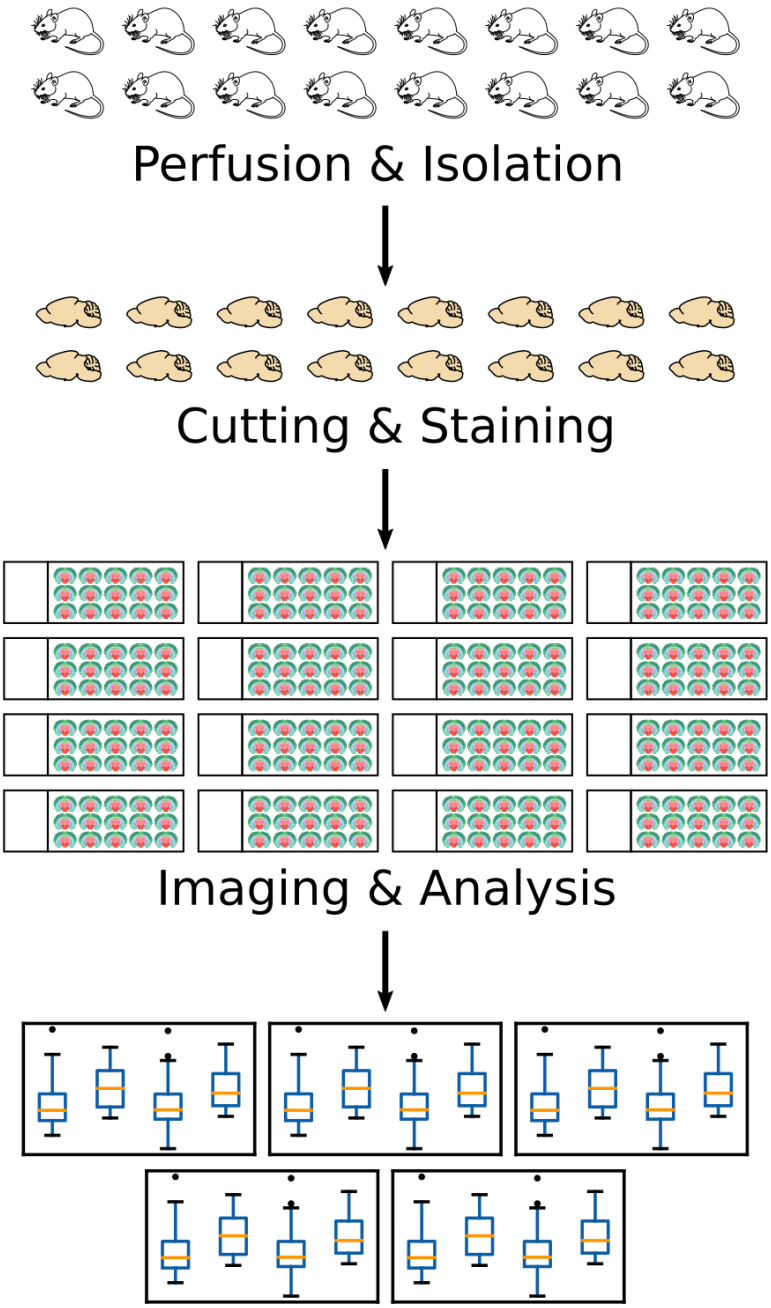


Figure 7: The main parts of the experiment.[10][19][6]
Perfusion and Isolation : 16 mice (see table 1 for more details) were perfused, stained with WGA and there brains isolated.
Cutting and staining : The 16 brains were cut into 30 μm sections and stained to be able to image the DNA, AQP-4 and astrocytes.
Imaging and Analysis : The sections were then imaged with LSCM and the fluorescent intensity of AQP-4 surrounding the blood vessels was quantified with ImageJ. The intensity profiles were then processed in Python and the resulting data plotted.

3.1 Animals

Mouse	Sex	Genotype	Time point
1	f	Per2flfl	ZT6
2	f	Per2flfl	ZT18
3	m	Per2flfl	ZT6
4	m	Per2flfl	ZT18
5	f	TPer2KO	ZT6
6	f	TPer2KO	ZT18
7	m	TPer2KO	ZT6
8	m	TPer2KO	ZT18

(a) First set of mice

Mouse	Sex	Genotype	Time point
9	f	Per2flfl	ZT6
10	f	Per2flfl	ZT18
11	m	Per2flfl	ZT6
12	m	Per2flfl	ZT18
13	f	TPer2KO	ZT6
14	f	TPer2KO	ZT18
15	m	TPer2KO	ZT6
16	m	TPer2KO	ZT18

(b) Second set of mice

Table 1: The mice used for the experiment. The time point indicates a which time the mouse was perfused.

16 mice (8 males and 8 females), aged 12 to 16 weeks were acquired from the University of Fribourg, Faculty of Science and Medicine. Half were TPer2KO mice. The other 8 mice were controls, having had the Per2 gene floxed. The animals then had their light/dark cycle shifted so that they were able to be perfused within a few hours of each other. The mice were housed in single cages in a 12:12 light:dark cycle with ad libitum food and water access.

3.2 Perfusion and isolation

1. The mouse was administered a lethal dose of anesthetic (80 mg/Kg ketamine and 0.33 mg/Kg medetomidine). After a few minutes the pedal reflex test was performed to determine consciousness. If there was still a reaction, the test was repeated until there was none. If the reaction persisted, an additional dose of anesthetic was administered.

2. Once the mouse was determined to be in the anaesthetic plane, the heart was exposed and perfused with a physiological sodium chloride solution (0.9%) for 2 min at a rate of 10 mL/min. Paling of the liver was observed as an indication of the required blood clearance.

3. The perfusion setup was then switched to the IHC solution containing wheat germ agglutinin, and the perfused for 1 min at a rate off 5 mL/min.

4. The perfusion ended with 2 min of paraformaldehyde solution at a rate of 10 mL/min.

5. The brain was then isolated and stored in the paraformaldehyde solution for 24h, then transferred to a 30% sucrose in PBS solution for 24h and finally frozen in an embedding medium (OCT) and stored at -20°C.

6. 30 μm sections were then cut between 0.7 mm to -2 mm from Bregma and stored at -20°C in antifreeze.

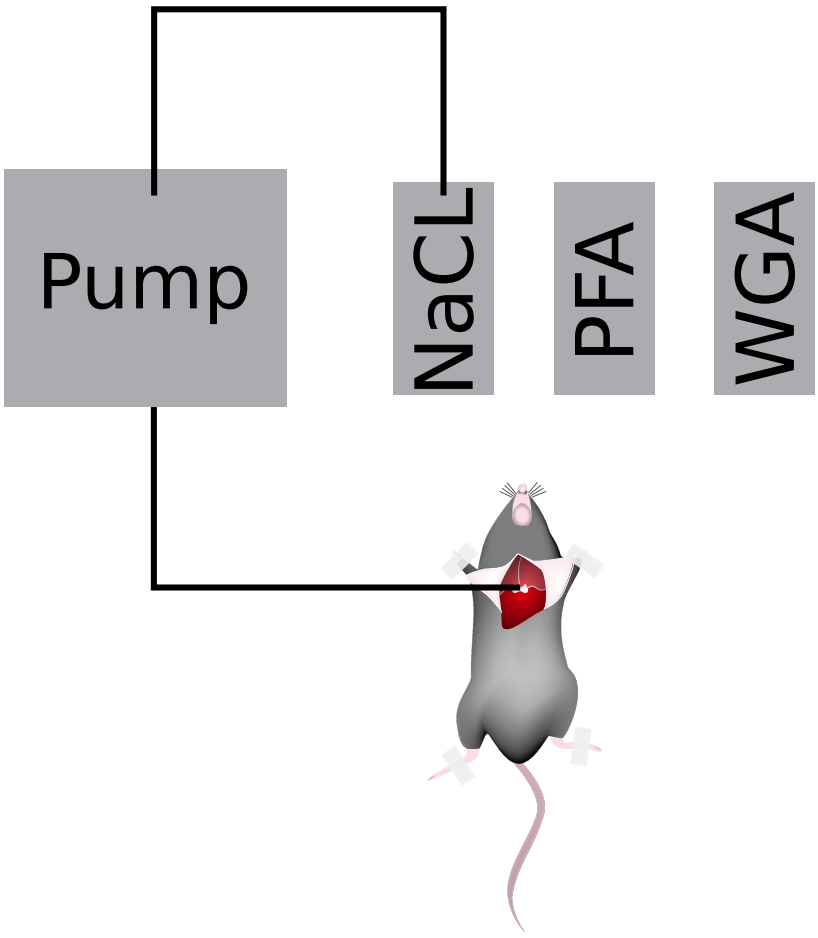


Figure 8: Illustration of the setup for perfusing the animals. NaCl : sodium buffer solution (0.9%), WGA : wheat germ agglutinin fluorescent staining solution, PFA : 4% paraformaldehyde in PBS fixing solution.[7]

3.3 Cutting and Staining

The brains were cut into 30 μm thick sections between 0.7 mm to -2 mm from Bregma with a Microm HM 550 Cryostat from GMI. The 30 μm thick brain sections were stained as follows :

1. Washed in TBS 3 times for 5 minutes at room temperature while shaking.
2. Blocked with 10% normal donkey serum in 1xTBS-T for 1 hour at room temperature while shaking. The TBS-Triton is used to increase the permeability of the cell membranes.
3. Incubated with anti AQP-4 and anti GFAP primary antibodies in 1% NDS overnight at 4 °C while shaking.
4. Washed in TBS 3 times for 5 min at RT while shaking.
5. Incubated with the fluorophore-linked secondary antibodies in 1xTBS-T and 1% NDS for 3 hours at room temperature while shaking.
6. Washed in TBS-T 3 times for 5 min at RT while shaking.
7. Stained with DAPI in 1xTBS-T for 10 minutes at room temperature while shaking.
8. Washed in TBS 3 times for 5 min at RT while shaking.
9. Mounted with Fluoromount G.

3.4 Microscopy

The microscopy used was the STELLARIS 8 FALCON from Leica. The images were acquired with either a resolution of 1024x1024 or 2048x2048 px at a rate of either 400 or 600 Hz with a 63x objective immersed in glycerine with a zoom factor of 0.75x and bit depth of 12. The microscope's 4 detectors were set to their photon counting mode. The 4 channels captured the emissions from the 4 fluorophores used during the perfusion and staining, DAPI (Em:461 nm), Alexa Fluor® 568 (Em:603 nm), Alexa Fluor® 647 (Em:667 nm) and Alexa Fluor® 488 coupled WGA (Em:516 nm), frame by frame. From the channel acquiring the data from the AQP4 staining, two frames were accumulated and averaged. 4 brain regions were imaged using the navigator feature of the LASX software : Dorsal cortex, Thalamus, Hypothalamus and the Suprachiasmatic nuclei. See figure 25 in the appendix for an image in the LASX software.

For each region one image composed of a 3x3 square (about 100x100 μm) was captured for each hemisphere.

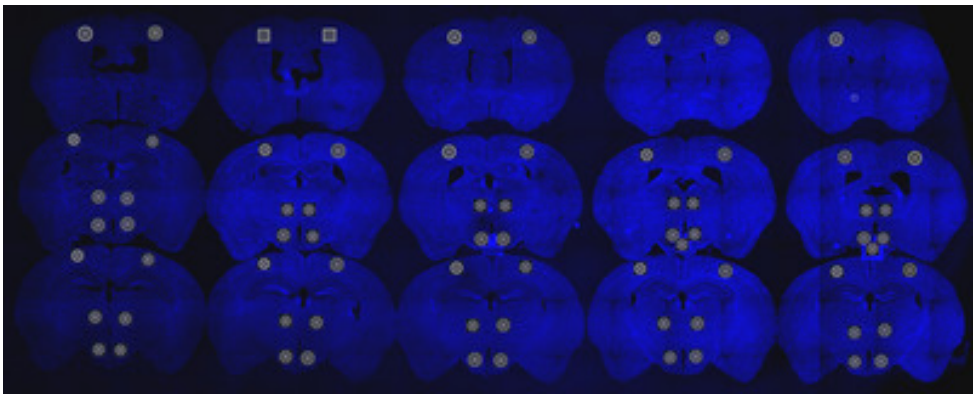


Figure 9: Image showing the layout of one of the 16 microscope slides. There are 15 sections, each 30 μm thick, between 0.7 mm to -2 mm from Bregma. The white regions show where the images will be acquired by the navigator feature of the LASX software. The staining is DAPI, captured at low resolution (256x256 px, 600 Hz) with a 5x objective.

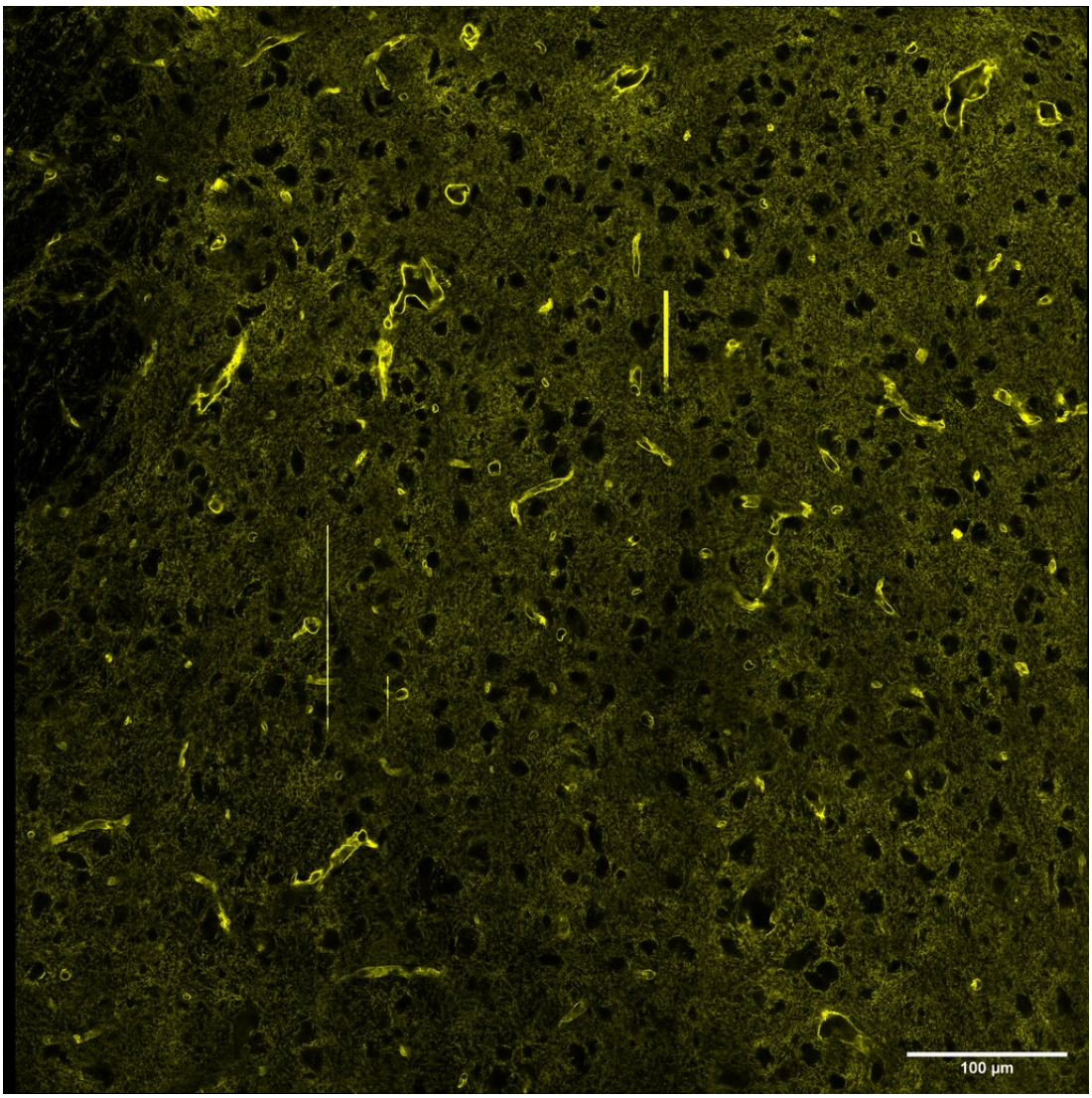


Figure 10: Image of part of the hypothalamus, acquired from animal 4 (control). Only the AQP-4 channel is shown.

3.5 Image analysis

One intensity profile from a 30 μm long line centered on a blood vessel with dimensions ranging from 5 to 8 μm in diameter and 20 μm in length was acquired from each of the images with the *plot profile* tool in FIJI(ImageJ).

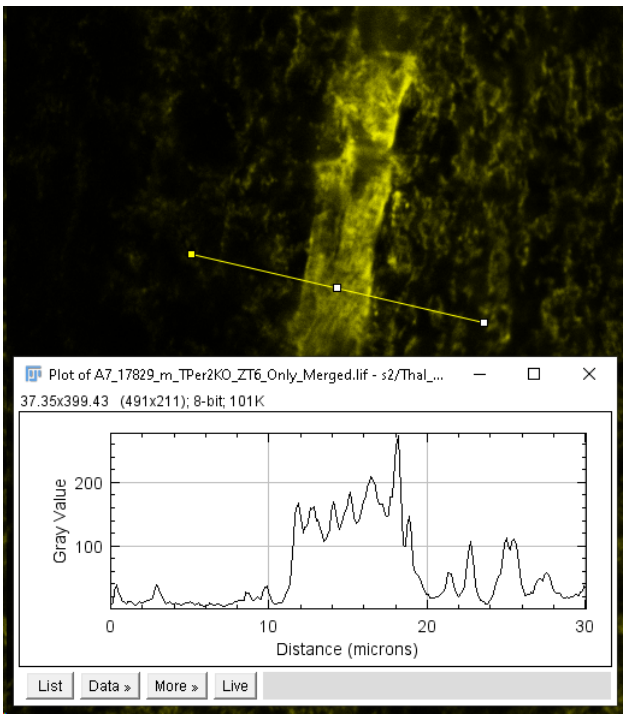


Figure 11: An example of an intensity profile. The yellow 30 μm long line centered on a blood vessel and the grey scale values provide the data for the line plot.

The profiles were then processed in Python :

1. The intensity profiles were separated by region, genotype, time point and sex.
2. The standard score (Z-score) of the intensity values was calculated.
3. The mean standard score values of the intensities for every 0.5 μm increment of the 30 μm long line was plotted in a line plot.
4. The maximum value standard score of each intensity profile was retrieved to make box plots comparing the different groups.

3.6 Statistical analysis

The hypothesis is that the quantity of the water transport channel AQP-4 is higher during the rest phase of circadian cycle, for mice this corresponds to the light phase. Therefore for the wild type mice, the measured relative intensity of the AQP4 staining should be higher for the vessels fixed at ZT6 I_{ZT6}, when compared to those fixed at ZT18 I_{ZT18} :

$$I_{ZT6} > I_{ZT18} \tag{3.1}$$

If this is not the case, then there should be no observable difference between the two time points :

$$I_{ZT6} \leq I_{ZT18} \tag{3.2}$$

- Null hypothesis $H_0 : I_{ZT6} = I_{ZT18}$
- Alternative hypothesis $H_1 : I_{ZT6} > I_{ZT18}$
- Significance level of 5% : $\alpha = 0.05$

The test used to compare the intensity values of the blood vessels of each of the different groups of mice was a two-way analysis of variance (ANOVA).

4 Results

4.1 Blood vessels

Table (3) Number of blood vessels acquired for the experiment, separated by type.

sex	genotype	timepoint	Vessels	nb of mice
f	**Per2flfl**	**ZT18**	146	2
		ZT6	100	2
	TPer2KO	**ZT18**	117	2
		ZT6	109	2
m	**Per2flfl**	**ZT18**	122	2
		ZT6	149	2
	TPer2KO	**ZT18**	80	2
		ZT6	127	2
Total			950	16

Table (4) Number of blood vessels acquired for the experiment, separated by region.

region	
DorCX	378
Hypo	244
SCN	96
Thal	232
Total	950

Table 4: The number of blood vessels used for the data analysis, separated by type. See the appendix for a more detailed table (table 7)

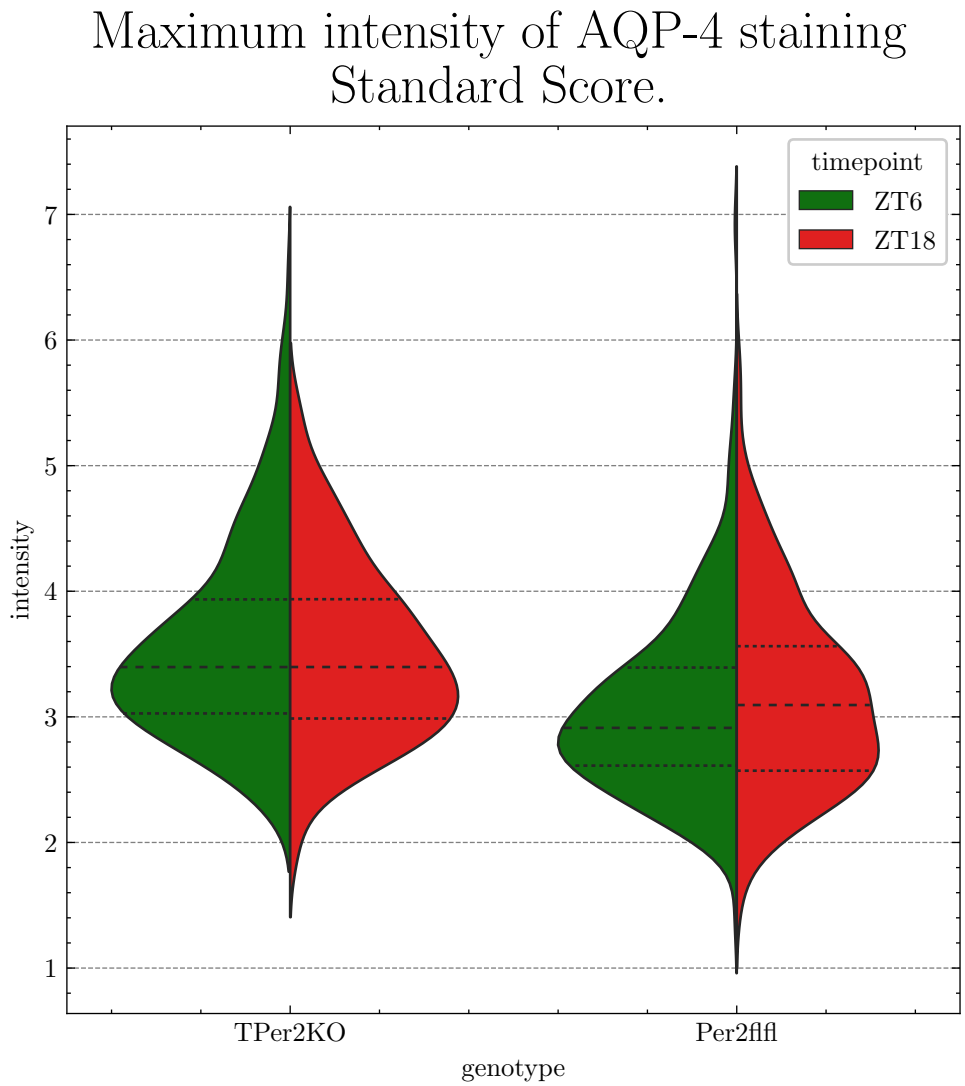
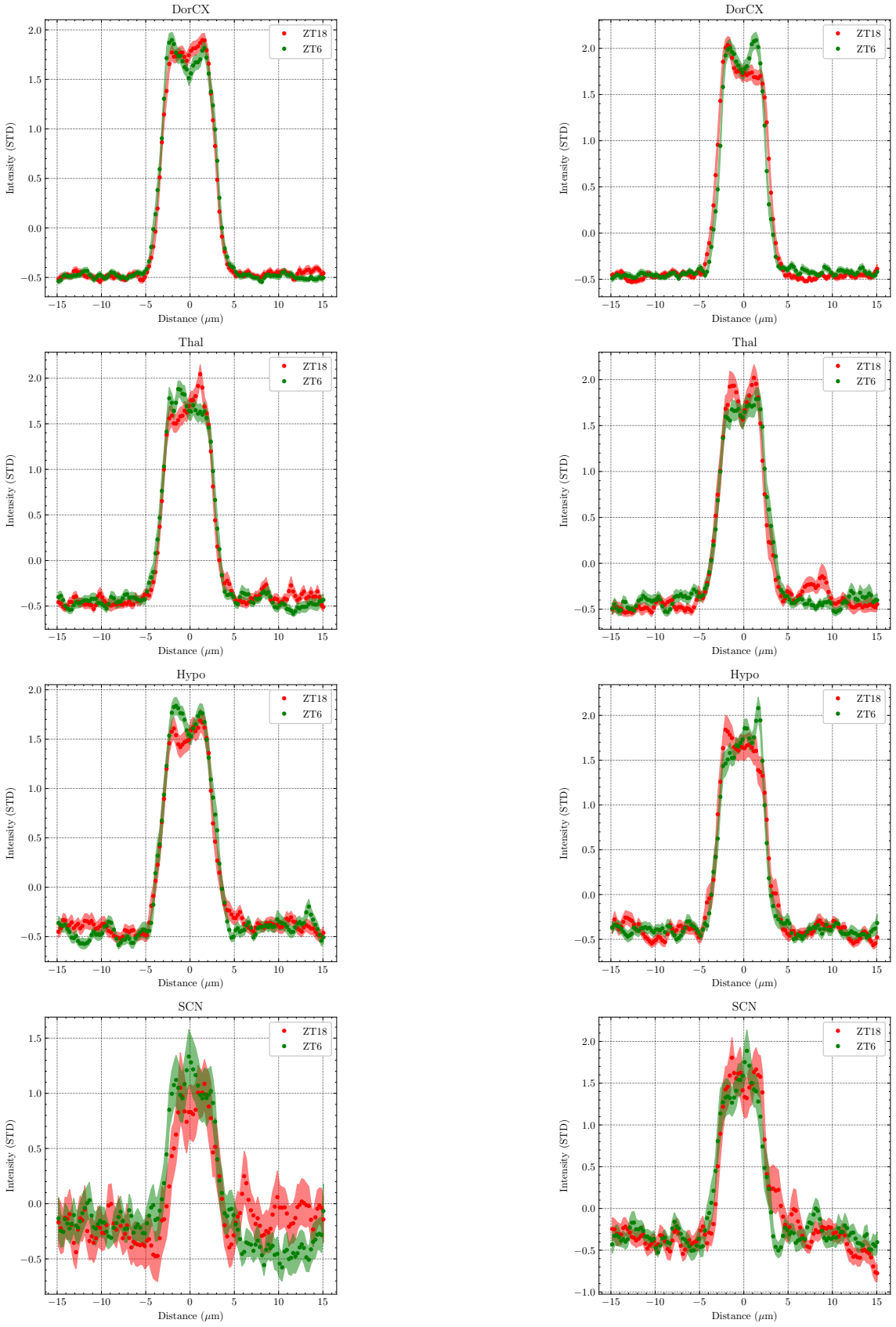


Figure 12: Violin plot visually comparing the influence of the time point and genotype on the intensity of the AQP-4 fluorescence. All of the separate brain regions taken together.

4.2 Line plots

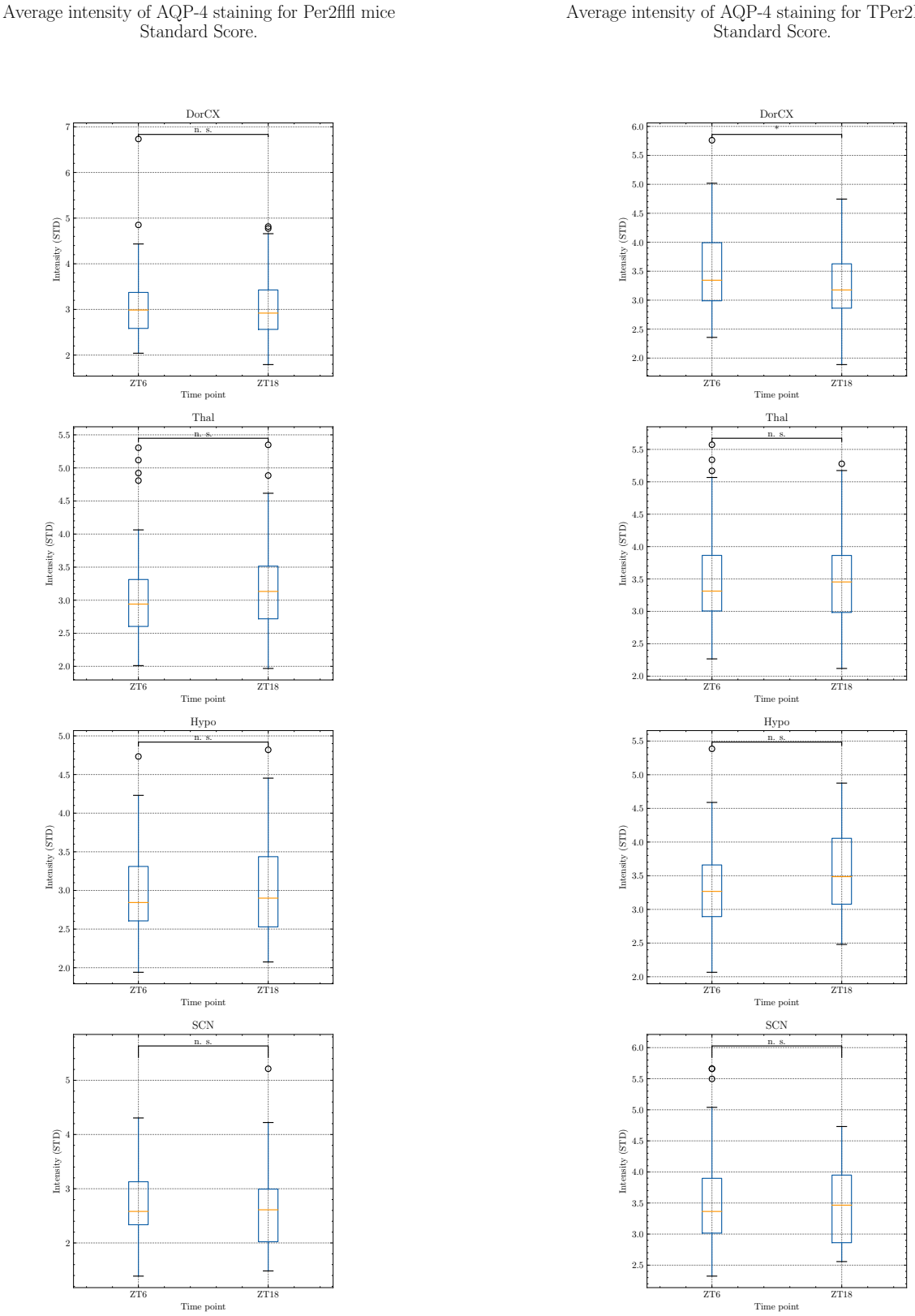


(a) Control mice

(b) Knockout mice

Figure 13: Line plot of the standard score of the intensity profiles of the control mice and mutant mice. The horizontal axis is centered in the middle of the blood vessel (see figure 11).

4.3 Box plots



(a) Control mice

(b) Knockout mice

Figure 14: Box plot of the maximum standard score of the intensity profiles of the control mice and mutant mice. See table 7 for the number of blood vessels per graph.

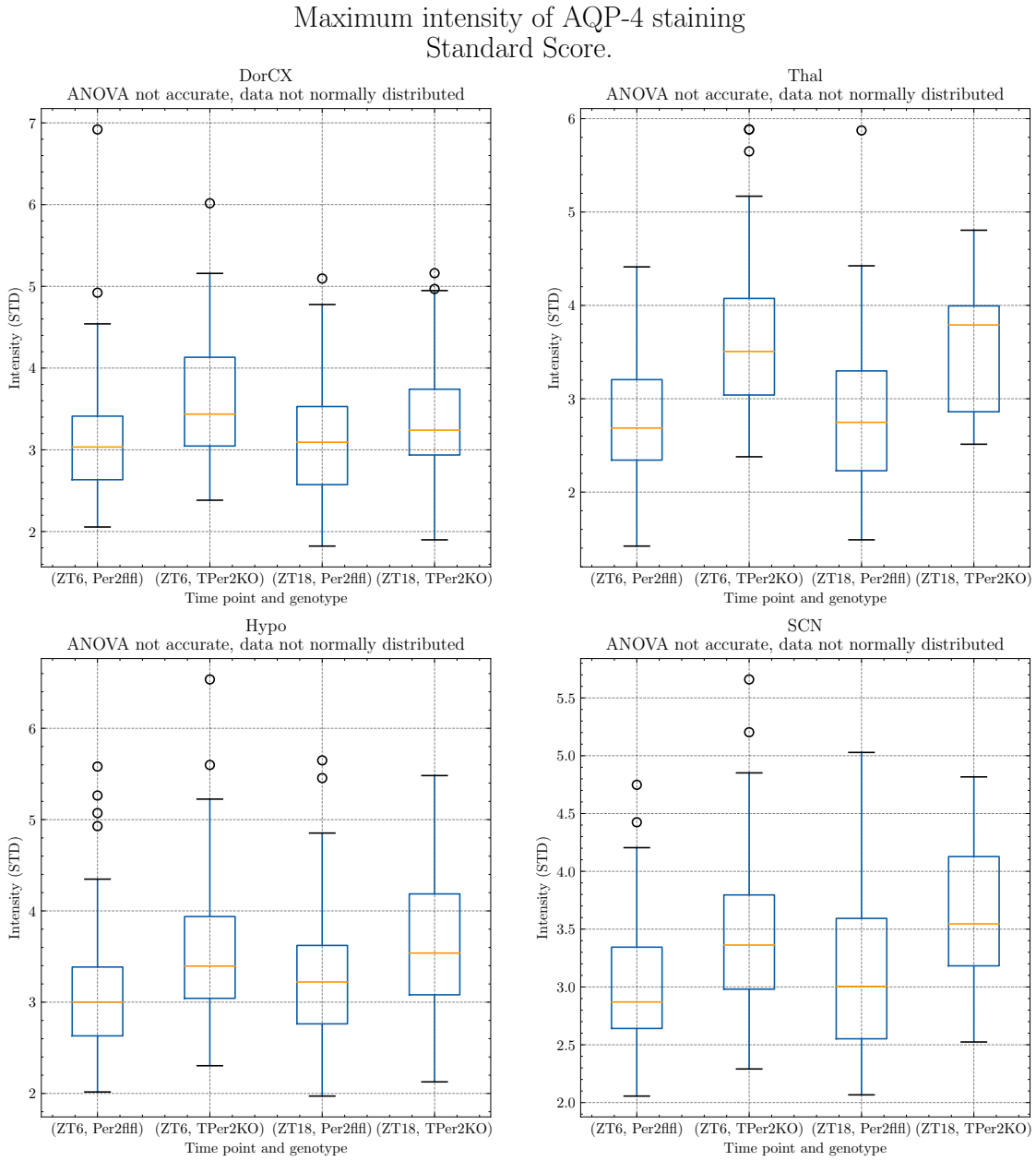


Figure 15: Box plot comparing the influence of the time point and genotype on the intensity of the AQP-4 fluorescence, separated by brain region. Outliers are represented as white circles.

Table 5: Resulsts of the ANOVA, separated by region.

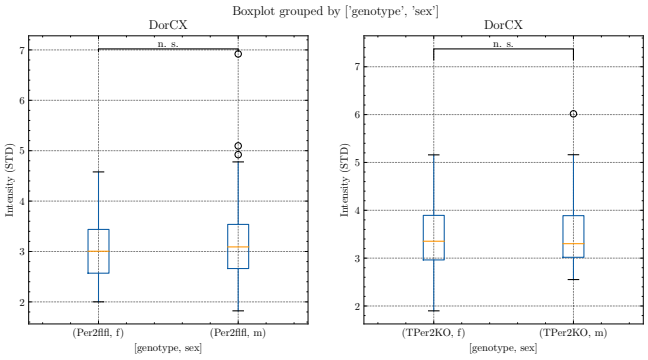
		Source	SS	DF	MS	F	p-unc	np2
DorCX	**0**	timepoint	0.7965	1.0000	0.7965	1.6109	0.2052	0.0043
	1	genotype	11.2692	1.0000	11.2692	22.7917	0.0000	0.0574
	2	timepoint * genotype	1.4621	1.0000	1.4621	2.9571	0.0863	0.0078
	3	Residual	184.9220	374.0000	0.4944	–	–	–
Thal	**0**	timepoint	0.1304	1.0000	0.1304	0.1704	0.6808	0.0018
	1	genotype	16.5366	1.0000	16.5366	21.6034	0.0000	0.1902
	2	timepoint * genotype	0.4454	1.0000	0.4454	0.5819	0.4475	0.0063
	3	Residual	70.4225	92.0000	0.7655	–	–	–

Continued on next page

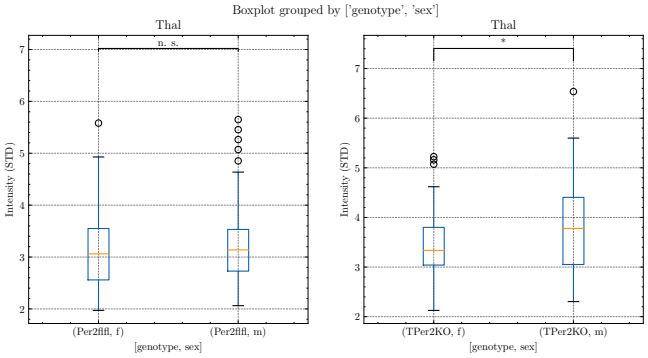
Table 5: Results of the ANOVA, separated by region.

		Source	SS	DF	MS	F	p-unc	np2
Hypo	**0**	timepoint	0.8726	1.0000	0.8726	1.3709	0.2429	0.0060
	1	genotype	10.1195	1.0000	10.1195	15.8991	0.0001	0.0652
	2	timepoint * genotype	0.3001	1.0000	0.3001	0.4714	0.4930	0.0021
	3	Residual	145.1179	228.0000	0.6365	–	–	–
SCN	**0**	timepoint	1.3675	1.0000	1.3675	3.1961	0.0751	0.0131
	1	genotype	12.3799	1.0000	12.3799	28.9336	0.0000	0.1076
	2	timepoint * genotype	0.1226	1.0000	0.1226	0.2865	0.5929	0.0012
	3	Residual	102.6894	240.0000	0.4279	–	–	–

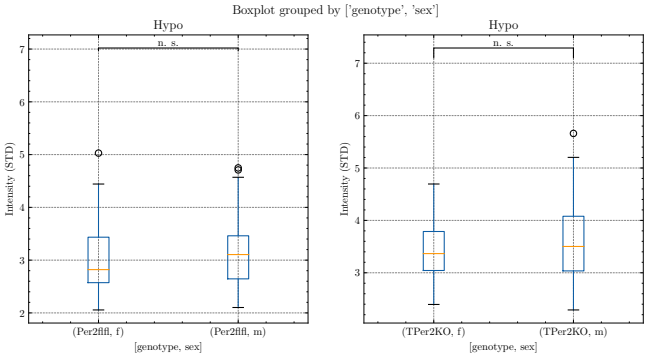
ANOVA summary:
'Source': Factor names
'SS': Sums of squares
'DF': Degrees of freedom
'MS': Mean squares
'F': F-values
'p-unc': uncorrected p-values
'np2': Partial eta-square effect sizes



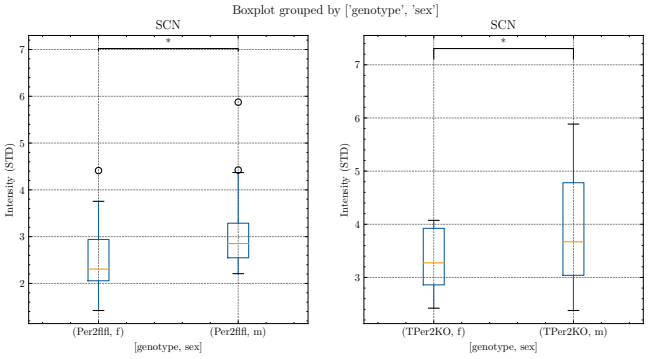
(a) Dorsal cortex



(b) Thalamus



(c) Hypothalamus



(d) Suprachiasmatic nuclei

Figure 16: Box plot comparing the influence of the sex on the intensity of the AQP-4 fluorescence, separated by brain region. t-test.

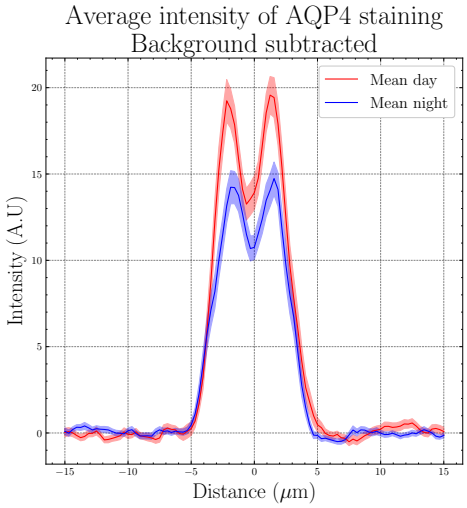
Table 6: Results of the t-test done between the sexes.

	DorCX	Thal	Hypo	SCN
statistic	0.746029	2.385209	1.391769	2.444517
pvalue	0.456772	0.019136	0.167136	0.018439

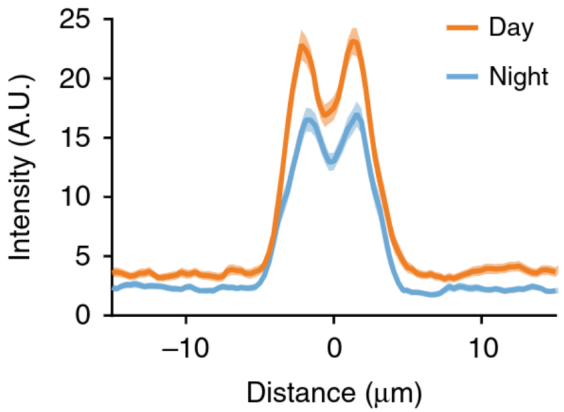
The blood vessel counts separated by brain region.

Region	
DorCX	378
Hypo	244
SCN	96
Thal	232
Total	950

4.4 Nedergaard data

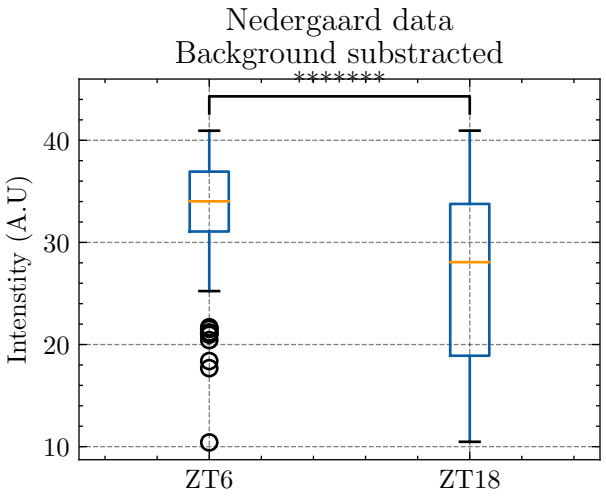


(a) Our data processing.

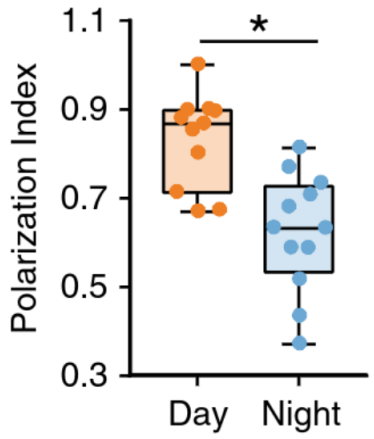


(b) Nedergaard's data processing. Adapted from Hablitz et al., 2020[11]

Figure 17: Nedergaard data set. Comparing the influence of the time point on the intensity of the AQP-4 fluorescence, background substracted.



(a) Our data processing.



(b) Nedergaard's data processing. Average polarization index boxplot. Polarization index equals peak vascular end foot fluorescence minus 10 μm baseline. All values were normalized to the highest signal for ease of visualization. Adapted from Hablitz et al., 2020[11]

Figure 18: Nedergaard data set. Comparing the influence of the time point on the intensity of the AQP-4 fluorescence, Background substracted.

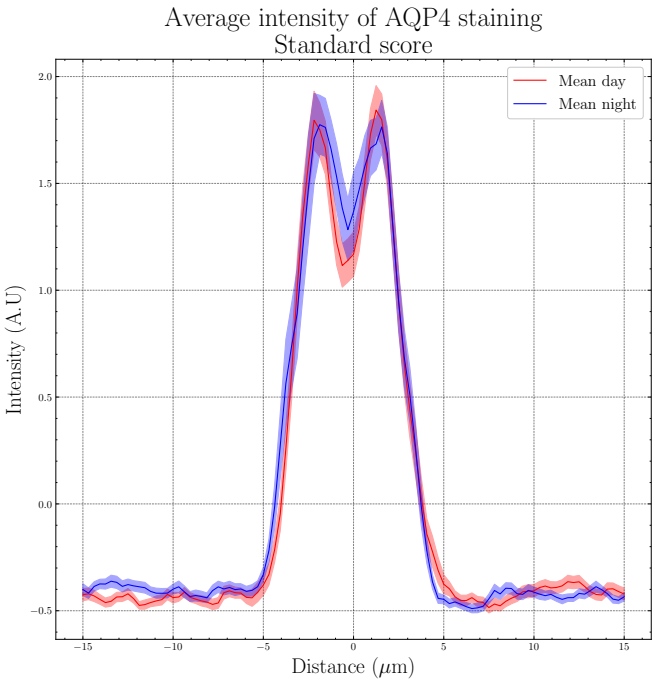


Figure 19: Nedergaard data set. Comparing the influence of the time point on the intensity of the AQP-4 fluorescence, standard score.

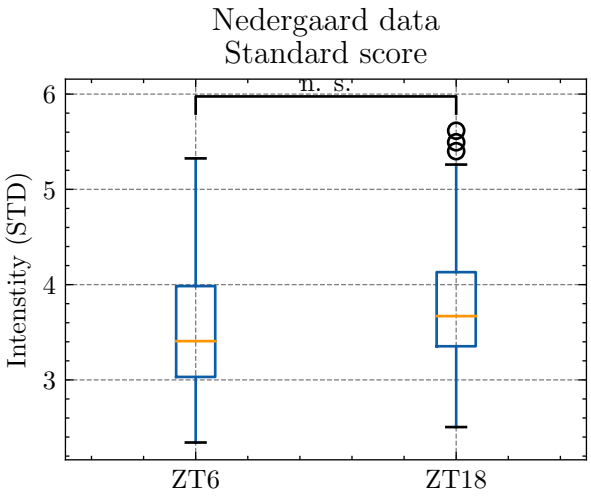
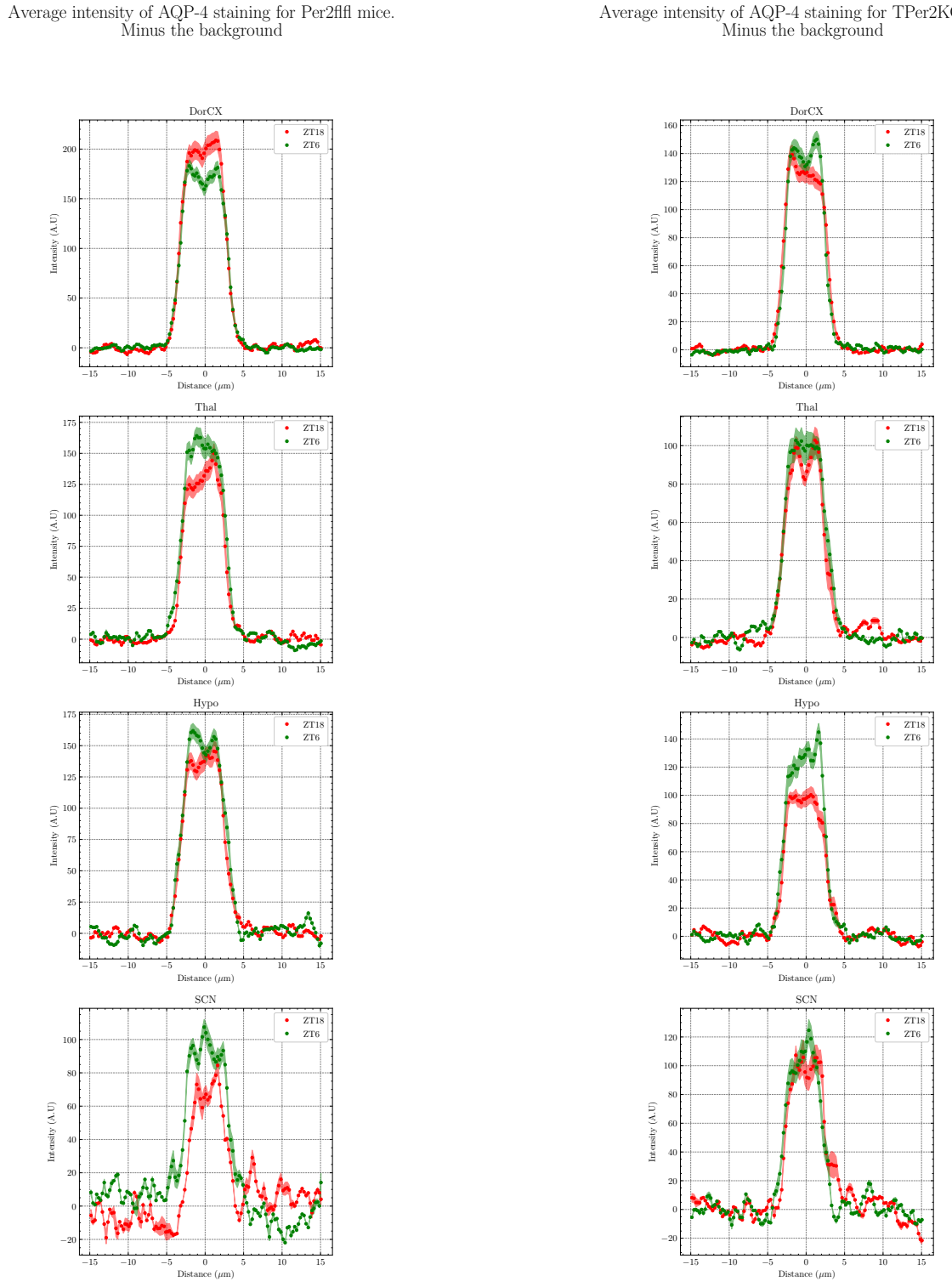


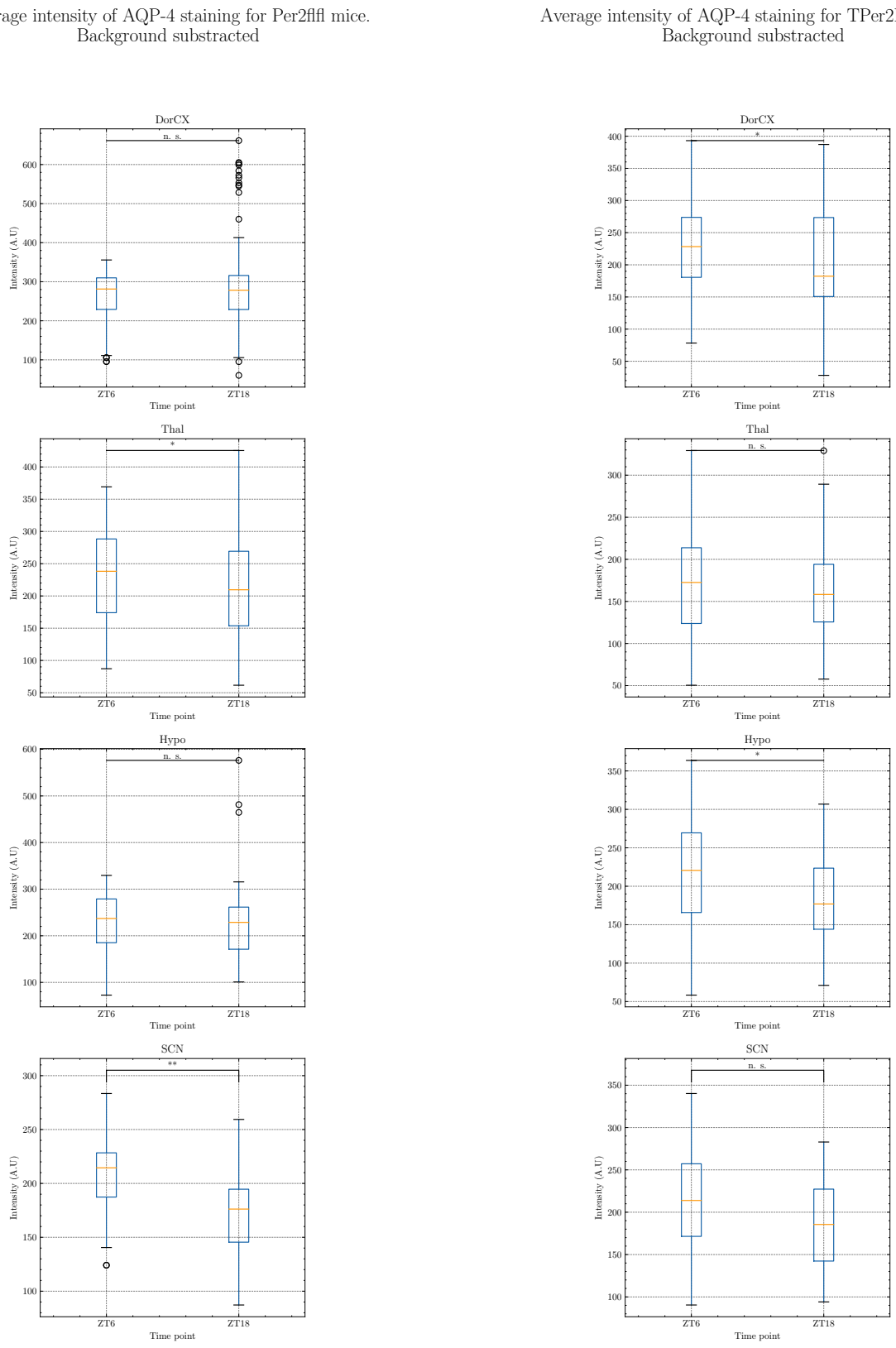
Figure 20: Nedergaard data set. Comparing the influence of the time point on the intensity of the AQP-4 fluorescence, standard score.

4.5 Background Subtraction



(a) Control mice (b) Knockout mice

Figure 21: Line plot of the of the intensity profiles of the control mice and mutant mice. The horizontal axis is centered in the middle of the blood vessel (see figure 11). The background was calculated from -15 to -5 and 5 to 15 μm and the mean value subtracted from the whole data set.



(a) Control mice

(b) Knockout mice

Figure 22: Box plot of the maximum value of the intensity profiles of the control mice and mutant mice. See table 7 for the number of blood vessels per graph. The background was calculated from -15 to -5 and 5 to 15 μm and the mean value subtracted from the whole data set.

5 Discussion

Figure 12 gives a good qualitative overview of the entire dataset. Looking at it, we can see that the data is not normally distributed, which unfortunately renders the ANOVA test useless. Nonetheless, we can see that the intensities of the control mice are slightly less than those of the KOs, which indicates a higher overall activity of the glymphatic system. Interestingly, the plot indicates that there is an increase of AQP-4 at ZT18, during the active phase of the mice, contrary to the original hypothesis.

There were technical difficulties during the imaging of mice 13-16, which resulted in less useable images than the rest of the mice. The data for the suprachiasmatic nuclei is not of very high quality and there are fewer actual blood vessels to image, since it is the smallest out of the 4 regions. Moreover, the vessels that were of sufficient image qualitative barely met the dimensional criteria, and were very difficult to make out on the image.

The line plots in section 4.2 have had their data manipulated : the intensity values have been merged in to bins of 0.5 μm. Since the operation was a simple averaging, this results in the introduction of a slight error. This error can be considered negligible, especially since these line plots serve only a qualitative purpose. The statistical tests were done on data that did not undergo the binning process. Visually, both genotypes appear to be very similar, with the notable feature being the increased intensity in the mutant mice. The box plots comparing the time points within the genotypes (figure 14) confirm this : no statistically significant differences were found.

Figure 15 is where the most conclusive statistical test of the experiment would have been done, but the data is not normally distributed, rendering the two-way ANOVA inconclusive. There was however a statistically significant result between the genotype, confirming the increase of AQP-4 intensity seen in figure 12.

Figure 16 confirms that the sex of the animal does not influence the fluorescent intensity of AQP-4. The statistically significant in the SCN can, be discarded due to the poor SCN data, as described above.

We only obtained similar results to the Nedergaard paper when comparing the dorsal cortex our control group to theirs : the intensity during the rest phase (ZT6) is inferior then during the active phase (ZT18). But when looking at the standard scores of the data, the seemingly obvious difference disappears. This highlights the fact that fluorescent microscopy is not a quantitative method ! The idea to take the standard score was proposed by a bioimage informatics expert, Felix Meyenhofer. The goal of the standard score is to be able to better compare data from different images, since the source of the images varies a considerable amount; different animals, different IHC staining sessions, etc. The Nedergaard team simply subtracted the background fluorescence, which is a much simpler approach. This method is not conclusive, as the apparent statistically significant different disappears when using the standard scores of the data.

Looking at the TPer2KOs, there were no statistically significant differences between the 2 time points, which would indicate that the perivascular polarization of AQP-4 is indeed effected by the circadian rhythm. But since the control group did not show statistically significant differences either when using the standard scores, this conclusion does not have a strong base. Since we cannot reproduce the day night difference with our control group, we must reconsider the statement regarding the circadian rhythm's control over the glymphatic system, or any other

day/night pattern. What the data does show however, is the seemingly increased perivascular polarization of AQP-4 expressed by the total Per2 knockouts compared to the control group.

6 Conclusion

The main challenge in confirming if the perivascular polarization of the water channel protein AQP-4 is effected by the circadian rhythm is the fact that, ideally, the animal should be alive during the quantification. Since we lack the necessary instruments to perform the experiment on a live test subject, we must make compromises. If we ignore the biological localization of the protein, id est only select for AQP-4 localised to the astrocytic endfeet, we can perform very accurate quantification by mass spectrometry or western blot. An other option is to quantify the AQP4 mRNA, but again this does not keep the localization information and also is not a direct measurement of the protein itself. Thus we are left with fluorescent microscopy. By fixing the brain tissue while the animal is still alive, we keep the spacial information of the brain structure. This allow us to reconstruct the blood vessels in 3 dimensions using a computer. Using the fluorescence of the molecule from the IHC staining, we can obtained semi-quantitative information about the protein of interest. While not as conclusive as MS, it can give a good basis for testing our hypothesis.

Our data and analysis leads us to the conclusion that the perivascular polarization of AQP-4 could be influenced by the circadian rhythm, but the experiments did not provide statistically significant proof. If we were to continue working on this project there are several ways to proceed :

- Acquire a larger dataset to perform the statistical analysis on and use more different tests, such as Kruskal–Wallis test.
- Automate the data acquisition from the images and improve the Python program to be able to accept less specific data and implement a GUI, which would facilitate the process for anyone to use this semi-quantitative method for their own experiments.
- Perfect the IHC staining during the perfusion step, and use a 3-Dimensional approach when quantifying the AQP-4 polarization. See figure 23.

(a) Graph (b) Reconstructed surface (c) Original Z-stack

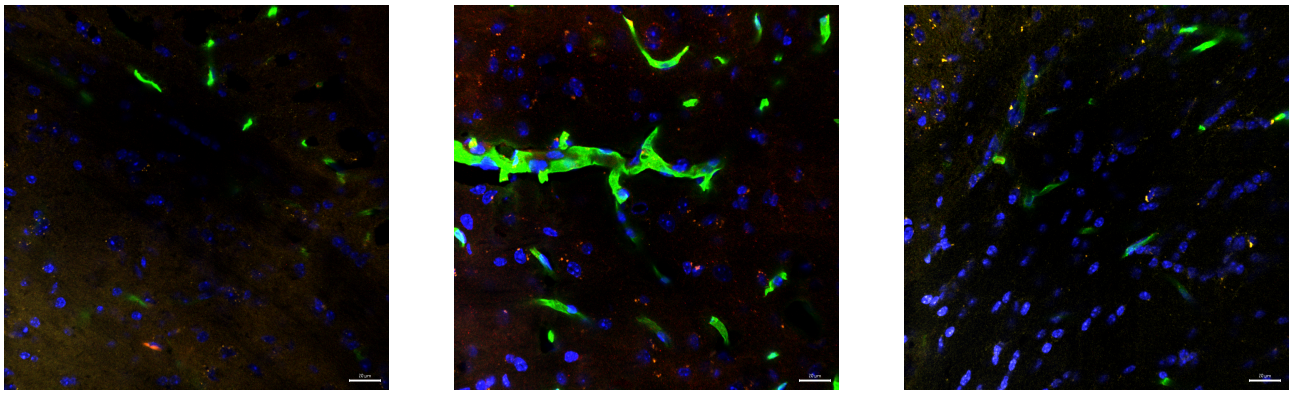
Figure 23: Left, graph of the mean intensity of AQP-4 as a function of the distance to the blood vessels. Middle and right show the images from which the data was plotted.

7 Acknowledgements

I would like to thank Urs Albrecht for welcoming me into his lab, Boris Egger and Felix Meyenhofer for sharing their expertise in microscopy and bioimage informatics, Stéphanie Aebischer for sharing her expertise on the cryostat and finally Katrin Wendrich for her excellent advice, guidance and help throughout my time in the lab and for designing the entire project.

8 Appendix

8.1 Antibody Controls



(a) Animals 1 and 2 (b) Animals 3 to 8 (c) Animals 9 to 16

Figure 24: Negative antibody controls for the IHC stainings. Green : blood vessels, blue : DNA and red and yellow for the astrocytes and AQP-4, respectively, are the negative controls.

8.2 Microscope Settings

Figure 25: STELLARIS microscope settings showing the setup for acquiring the images.

8.3 Extra blood vessel data

Table 7: Number of blood vessels acquired for the experiment, separated by type.

sex	genotype	timepoint	region	0
f	Per2fl/fl	ZT18	DorCX	56
			Hypo	42
			SCN	10
			Thal	38

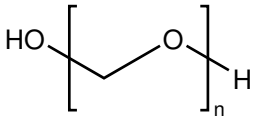
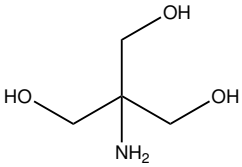
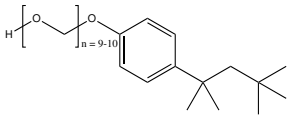
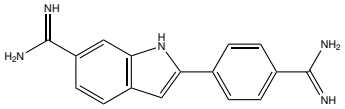
Continued on next page

Table 7: Number of blood vessels acquired for the experiment, separated by type.

sex	genotype	timepoint	region	0
		ZT6	**DorCX**	43
			Hypo	24
			SCN	12
			Thal	21
	TPer2KO	**ZT18**	**DorCX**	53
			Hypo	26
			SCN	14
			Thal	24
		ZT6	**DorCX**	49
			Hypo	28
			SCN	6
			Thal	26
m	**Per2flfl**	**ZT18**	**DorCX**	49
			Hypo	31
			SCN	10
			Thal	32
		ZT6	**DorCX**	55
			Hypo	39
			SCN	15
			Thal	40
	TPer2KO	**ZT18**	**DorCX**	36
			Hypo	19
			SCN	6
			Thal	19
		ZT6	**DorCX**	37
			Hypo	35
			SCN	23
			Thal	32
Total				950

8.4 Chemicals

Table 8: Chemicals used for the mice perfusion and immunohistochemistry

Name	Structure	Molecular weight, g/mol	Toxicity
Paraformaldehyde		30.03 (as monomer)	
Sodium hydroxide	NaOH	40.00	
Sodium phosphate			
Sodium chloride	NaCl	58.44	–
Tris(hydroxymethyl) aminomethane		121.136	–
Hydrochloric acid		36.46	
Sodium chloride	NaCl	58.44	–
Triton X-100		647	
DAPI		277.331	

8.4.1 Stock solutions for perfusion

Table 9: Stock solution preparation for the mice perfusions

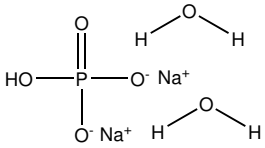
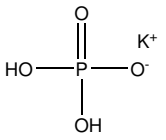
10x Phosphate-buffered saline (PBS), pH=7.4, autoclaved				**Volume: 1 L**	
Name	**Structure**	**Molar mass, g/mol**	**Amount**	**Final concentration**	**Toxicity**
Sodium chloride	NaCl	58.44	80 g	1.4 M	–
Potassium chloride	KCl	74.555	2 g	27 mM	–
Disodium hydrogen phosphate dihydrate		177.99	14.2 g	80 mM	–
Potassium dihydrogen phosphate		136.086	2 g	15 mM	–
Distilled water	H_2O	18	ca. 800 mL	–	–
Notes: pH to 7.4, q.s with dH_2O to 1 L, autoclave (liquid cycle)					

Table 10: Stock solution preparation for the mice perfusions

5 M NaCl				**Volume: 0.5 L**	
Name	**Structure**	**Molar mass, g/mol**	**Amount**	**Final concentration**	**Toxicity**
Sodium chloride	NaCl	58.44	146.1 g	5 M	–
Distilled water	H_2O	18	ca. 450 mL	–	–
Notes: q.s with dH_2O to 0.5 L, store at room temperature					

Table 11: Stock solution preparation for the mice perfusions

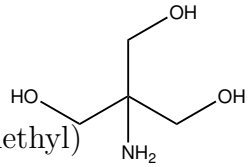
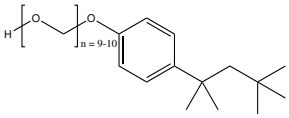
10x Tris-buffered saline (TBS), pH7.5					**Volume: 1 L**
Name	**Structure**	**Molar mass, g/mol**	**Amount**	**Final concentration**	**Toxicity**
Sodium chloride	NaCl	58.44	87.7 g	1.5 M	–
Tris(hydroxymethyl) aminomethane		121.14	121.1 g	1 M	–
Distilled water	H_2O	18	ca. 800 mL	–	–
Notes: q.s with dH_2O to 1 L, pH to 7.5, store at RT					
1x TBS-Triton (TBS-t))					**Volume: 1 L**
Name	**Structure**	**Molar mass, g/mol**	**Amount**	**Final concentration**	**Toxicity**
10x TBS	H_2O	18	ca. 800 mL	–	–
Triton X-100		647	1 mL	0.1%	
Distilled water	H_2O	18	900 mL	–	–
Notes:					

Table 12: Preparation of Fixative and Buffers for perfusion.[9]

Prepare 8% Paraformaldehyde stock
Add 40 g Paraformaldehyde to 500 ml dH_2O. Heat the solution to 60-65 °C while stirring (Do not exceed 65 °C, doing so can adversely affect the success of the immunohistochemical procedure).
To clear the solution, reduce heat and add 2-3 ml of 1.0 M NaOH with a dropper.
Filter and store at 4 °C for up to 1 month.
Prepare 0.2 M Sodium Phosphate Buffer, pH 7.4
For the sodium phosphate monobasic stock, add 27.8 g NaH2PO4* H_2O to 1 L dH_2O.
For the sodium phosphate dibasic stock, add 28.4 g Na2HPO4 to 1 L dH2O.
Add 810 ml of the monobasic stock to 190 ml of the dibasic stock.
Prepare 4% Paraformaldehyde Fixative
Add equal parts 8% paraformaldehyde stock to 0.2M Sodium Phosphate Buffer
Note: this fix is best prepared fresh, no more than 72 hours in advance.

8.4.2 Antibodies

Table 13: Primary Antibodies for the immunohistochemical staining

Protein	**Species Reactivity**	**Host**	**Antibody Type**	**Concentration**	**Producer**
Anti-Aquaporin 4 Antibody, CT	Human, Mouse, Rat	Rabbit	Polyclonal	1:500	merckmillipore[17]
Anti-GFAP Antibody	Human, Mouse, Rat	Goat	Polyclonal	1:500	abcam[1]

Table 14: Secondary Antibodies for the immunohistochemical staining

Specificity	**Host**	**Fluorophore**	**Concentration**	**Producer**
Anti-rabbit IgG	Donkey	Alexa Fluor® 568 Ex:578, Em:603 nm	1:500	abcam[2]
Anti-goat IgG	Donkey	Alexa Fluor® 647 Ex:651, Em:667 nm	1:500	Jackson ImmunoResearch[15]

8.5 Code

```
macro "Fixed Length Line Tool [f1]" {
  var desiredLength = 30; //in scaled units
  var isMaxlength = false; //not fixed, only maximum line length
  var isCentered = true;

  leftClick = 16;
  shift = 1;
  getPixelSize(unit, pixelWidth, pixelHeight);
  getCursorLoc(x0, y0, z, flags);
  lastX = x0;
  lastY = y0;
  dx = (x0);
  dy = (y0);
  dxs = dx * pixelWidth;       //in scaled units
  dys = dy * pixelHeight;
  length = sqrt(dxs*dxs + dys*dys);
  if (isCentered)
    length = 2*length;
  enlageFactor = desiredLength/length;
  if (isMaxlength && enlageFactor > 1)
    enlageFactor = 1;
  if (isCentered) {
    xS = x0 - dx*enlageFactor;
    yS = y0 - dy*enlageFactor;
  } else {
    xS = x0;
    yS = y0;
  }
  x = x0 + dx*enlageFactor;
  y = y0 + dy*enlageFactor;
  makeLine(xS, yS, x, y);
}

macro "Send To Excel [f2]" {
  //====================================
  // Macro that appends the data from the "Analyze --> Plot Profile" tool
  // to an excel file in the same directory as the current image
  // Requires the "Read and Write Excel" Plugin :
    // Update Site :
    // https://sites.imagej.net/ResultsToExcel/
  // Links :
  // https://imagej.net/Read_and_Write_Excel
  // https://imagej.github.io/plugins/read-and-write-excel
  // https://github.com/bkromhout/Read_and_Write_Excel_Modified
  //
  // An line tool macro is very useful, such as :
  // https://wsr.imagej.net/plugins/Tools/Fixed_Length_Line_Tool.ijm
  //====================================

  // get the results
  run("Clear Results");
  getLine(x1, y1, x2, y2, lineWidth);
  if (x1==-1)
    exit("This macro requires a straight line selection");
  run("Plot Profile");
  Plot.getValues(xpoints, ypoints);
  run("Close");

```

```
  // Copy data to the results window for the read and write excel fct
  // Substract half of the length to center the distance around the middle of the line
  getPixelSize(unit, pw, ph);
  x1*=pw; y1*=ph; x2*=pw; y2*=ph;
  dx = x2-x1; dy = y2-y1;
  length = sqrt(dx*dx+dy*dy); // length of the measuring line
  for (i=0; i<xpoints.length; i++)
    setResult("Distance", i, xpoints[i]-(length/2));
    for (i=0; i<ypoints.length; i++)
    setResult("Value", i, ypoints[i]);

  // Same path and name_DATA as the current image
  excel = ".xlsx";
  txt = "_DATA";
  path = getInfo("image.directory");
  title = getTitleStripExtension();
  title = title + txt + excel;
  file = path + title; // String to give to read and write excel fct

  // Adds the data to the the excel file defined as "file" above
  run("Read and Write Excel", "no_count_column file="+file);


  //====================================
  // Use this function to strip any number of extensions
  // off images.
  // Returns the title without the extension.
  // Auther : Michael Cammer
  //====================================
  function getTitleStripExtension() {
    t = getInfo("image.filename");
    t = replace(t, ".tif", "");
    t = replace(t, ".tiff", "");
    t = replace(t, ".lif", "");
    t = replace(t, ".lsm", "");
    t = replace(t, ".czi", "");
    t = replace(t, ".nd2", "");
    return t;
  }
}
```

Listing 1: ImageJ macros for saving the intensity profiles from the microscope images.

```
# "This script will make a DataFrame from all the .xlxs files in the given directory a

# Dependencies:

# Pandas, Matplotlib, Numpy, Scipy, Pingouin (conda install -c conda-forge pingouin),

#TODO add logging

"""
  Using the line plot tool form imageJ and the send to excel plugin, we can make
  an excel file that contains the data from the plot profile tool. There is one
  excel file per .lif file, containing the data from all the images in that .lif.
  You may choose the regions, genotype and the sex that appear in the plot, so
  long as you know the naming scheme used when naming the images acquired during the m
"""

import os
import sys
```

```
import warnings
import logging

__author__ = "David Parker"
__version__ = "1.0.0"
__title__ = "Plot_All"
__license__ = "GPLv3"
__author_email__ = "david.parker@unifr.ch"

def parseArgs():
  """Parse command line arguments"""

  import argparse

  try:
    parser = argparse.ArgumentParser(
        description='Make plots from excel files in directory')

    parser.add_argument('-p',
              '--path',
              action='store',
              required=True,
              help='The path to the directory with the excel files.')
    parser.add_argument('-v',
              '--verbose',
              action='count',
              default=0,
              help='Verbose behaviour, printing parameters of the script.')
    parser.add_argument('-r',
              '--regions',
              nargs="*", # 0 or more values expected => creates a list
              type=str,
              default=['DorCX','Thal', 'Hypo', 'SCN'],
              help='List with the desired regions to plot. Defaults to : DorCX Thal Hy
    parser.add_argument('-g',
              '--genotypes',
              nargs="*",
              type=str,
              default=['TPer2KO', 'Per2flfl'],
              help='The genotypes to plot. Choose from Per2flfl or TPer2KO')
    parser.add_argument('-s', #TODO add this
              '--sex',
              action='store',
              default=False,
              help='The sex you wish to plot. Choose from f or m. Leave blank to combi
    parser.add_argument('-x',
              '--excel',
              action='store_true',
              #default=False,
              help='Choose to write the different stages of the data processing to exc
    parser.add_argument('-t',
              '--tex',
              action='store_true',
              #default=False,
              help='Choose to write the different stages of the data processing to tex
  except:
    print("An exception occurred with argument parsing. Check your provided options.")
    traceback.print_exc()

  return parser.parse_args()
```

```
79
80 # Place all the functions here
81
82 # MATLAB's tic toc equivalent (From user:4191389 Benben on SO)
83 import time
84 def TicTocGenerator():
85   # Generator that returns time differences
86   ti = 0           # initial time
87   tf = time.time() # final time
88   while True:
89     ti = tf
90     tf = time.time()
91     yield tf-ti # returns the time difference
92 TicToc = TicTocGenerator() # create an instance of the TicTocGen generator
93 # This will be the main function through which we define both tic() and toc()
94 def toc(tempBool=True):
95   # Prints the time difference yielded by generator instance TicToc
96   tempTimeInterval = next(TicToc)
97   if tempBool:
98     print( "Elapsed time: %f seconds.\n" %tempTimeInterval )
99 def tic():
100   # Records a time in TicToc, marks the beginning of a time interval
101   toc(False)
102
103 # (From user:8394915 cheersmate on SO)
104 def barplot_annotate_brackets(num1, num2, data, center, height, ax, yerr=None, dh=.05,
105   """
106   Annotate barplot with p-values.
107   Ty cheersmate from SO
108   https://stackoverflow.com/questions/11517986/indicating-the-statistically-significan
109
110   :param num1: number of left bar to put bracket over
111   :param num2: number of right bar to put bracket over
112   :param data: string to write or number for generating asterixes
113   :param center: centers of all bars (like plt.bar() input)
114   :param height: heights of all bars (like plt.bar() input)
115   :param yerr: yerrs of all bars (like plt.bar() input)
116   :param dh: height offset over bar / bar + yerr in axes coordinates (0 to 1)
117   :param barh: bar height in axes coordinates (0 to 1)
118   :param fs: font size
119   :param maxasterix: maximum number of asterixes to write (for very small p-values)
120   """
121
122   import matplotlib.pyplot as plt
123
124   if type(data) is str:
125     text = data
126   else:
127     # * is p < 0.05
128     # ** is p < 0.005
129     # *** is p < 0.0005
130     # etc.
131     text = ''
132     p = .05
133
134     while data < p:
135       text += '*'
136       p /= 10.
137
138       if maxasterix and len(text) == maxasterix:
```

```
        break

    if len(text) == 0:
      text = 'n. s.'

  lx, ly = center[num1], height[num1]
  rx, ry = center[num2], height[num2]

  if yerr:
    ly += yerr[num1]
    ry += yerr[num2]

  ax_y0, ax_y1 = plt.gca().get_ylim()
  dh *= (ax_y1 - ax_y0)
  barh *= (ax_y1 - ax_y0)

  y = max(ly, ry) + dh

  barx = [lx, lx, rx, rx]
  bary = [y, y+barh, y+barh, y]
  mid = ((lx+rx)/2, y+barh)

  ax.plot(barx, bary, c='black')

  kwargs = dict(ha='center', va='bottom')
  if fs is not None:
    kwargs['fontsize'] = fs

  ax.text(*mid, text, **kwargs)

def fct_name(var1, var2, verbose):
  """Fct that checks the input data"""
  # import packages

  # Do a little sanity checking:

  if verbose > 0: print("what the fct/loop is doing")

  # check your input data
  a = 1
  if a != 1:
    sys.stderr.write("data makes no sense, check your data")
    sys.exit(1)

  return a

def make_big_DataFrame(path, verbose):
  """
    Fct that makes one big DataFrame with all the excel files in the directory.
  """
  import os
  import warnings
  import pandas as pd
  import traceback

  if verbose > 0: print("################################################\n")
  if verbose > 0: tic()

  big_df = []
  with os.scandir(path=path) as it:
```

```
    if verbose > 0: print('Scanning directory at ' + path +'\n')
    for entry in it:
      if entry.name.endswith(".xlsx") and entry.is_file():
        if verbose > 0: print('\tFound .xlsx file : importing data from', entry.name)
        # Import data
        warnings.simplefilter("ignore") # To suppress an annoying openpyxl warning
        df = pd.read_excel(path + entry.name,
          header=[0,1],
          dtype='float64')
        warnings.simplefilter("default")
        # Clean data
        if verbose > 0: print("\t\tRemoving empty columns")
        df.dropna(axis=1, how='all', inplace=True) # Remove empty columns
        if verbose > 0: print("\t\tAppending to master DataFrame\n")
        big_df.append(df)
  if verbose > 0: toc()
  if verbose > 0: print("################################################\n")

  return pd.concat(big_df, axis=1, verify_integrity=True, copy=False)

def merge_on_distance(df, verbose, bins):
  """
    Fct that takes all the unique (distance, value) pairs and
    makes a dataframe with one distance column containing all the
    unique distances, and then all the values as the rest of the
    columns, lined up to their corresponding distance.
    Many thanks to /u/sarrysyst

    An alternative is to use numpy.interp
    https://numpy.org/doc/stable/reference/generated/numpy.interp.html
  """

  import pandas as pd
  import numpy as np
  from functools import reduce

  if verbose > 0: tic()
  if verbose > 0: print("################################################\n")
  if verbose > 0: print("Beginning merging on distance process :")

  # Make list with the data split by level 0 header
  if verbose > 0: print("\tSplitting dataframe by image into list")
  df_list_small = []
  names = list(set(df.columns.get_level_values(0).tolist()))
  for name in names:
    df_list_small.append(pd.concat({name: df[name]}, axis=1).dropna(how='all', axis=0)
    if verbose > 1: print("\t\t", name)
  # get all unique distances into a separate dataframe
  if verbose > 0: print("\tRetrieving and sorting unique distances")
  # The np.sort step is CRUCIAL !
  unique_dists = pd.unique(df.filter(regex='Dist').values.ravel('K')) #.ravel makes a
  df_distances_big = pd.DataFrame({'dist':np.sort(unique_dists)})
  df_merged_small_list = [df_distances_big]

  # Start by merging all the smaller dfs in the list on their distances
  if verbose > 0: print("\tMerging dataframes in list")
  for df_small in df_list_small:

    # distance columns need to have the same column names to use merge later on
    col_names = []
```

```
    for i, col in enumerate(df_small.columns):
      if i % 2 == 0:
        col_names.append('dist')
      else:
        col_names.append(col)
    df_small.columns = col_names

    # get all unique distances into a separate dataframe
    df_distances_small = pd.DataFrame(df_small.iloc[:,::2].stack().unique(), columns=[
    # create list of dataframe slices of each individual dist/val pairs
    pairs_small = [df_distances_small] + [df_small.iloc[:,i:i+2] for i in range(0, len

    # merge individual slices on df_distances which has all unique distance values
    df_merged_small = reduce(lambda  left, right: pd.merge(left, right, on=['dist'],

    # this part is optional to reduce the distances into set bins, adjust the range an
    df_resampled_small = df_merged_small.groupby(pd.cut(df_merged_small["dist"], np.ar
    df_resampled_small.drop(columns=['dist'], axis=1, inplace=True)

    # Append the merged df to the list
    df_merged_small_list.append(df_merged_small.dropna(how='all', axis=0))

  # Not sure if I need to leave this line..
  df_master_list = df_merged_small_list.insert(0,df_distances_big)

  # merge the smaller merged dfs onto the sorted list of unique distances
  if verbose > 0: print("\tMerging merged dataframes on unique distances into masterfr
  df_merged_big = reduce(lambda  left, right: pd.merge(left, right, on=['dist'],  how=

  #df_merged_big.to_excel('NotBineed_Raw_data.xlsx')

  # this part is optional to reduce the distances into set bins, adjust the range and
  # !!! This introduces error : taking the mean of the values in the bin is not good,
  #     some columns are modified, while others aren't
  df_resampled_big = df_merged_big.groupby(pd.cut(df_merged_big["dist"], np.arange(-15
  # this doesnt work : df_resampled_big.drop(columns=['dist'], axis=1, inplace=True)

  if verbose > 0: print("\n")
  if verbose > 0: toc()
  if verbose > 0: print("\n##################################################\n")

  if bins :
    return df_resampled_big #this one has bins
  if not bins :
    return df_merged_big # no bins

def make_std_long(df, verbose, path, write_excel, tex):
  '''
  This takes the output of the merge_on_distance function, and processes the data into
  !!! It takes only the maximum standard score from each line profile !!!
  You must also have used a very specific file naming convention when aquiring the dat
    file name : "XXX_sex_genotype_timepoint_XXX.lif ", in that order
    Region names in the LasX navigator : region_XXX
  The exact names for each variable must be :
    sex : "m" or "f"
    genotype : "Per2flfl" or "TPer2KO"
    timepoint : "ZT6" or "ZT18"
    region names : "DorCX" or "Thal" or "Hypo" or "SCN"
  '''

```

```
import pandas as pd
import numpy as np

if verbose > 0: tic()
if verbose > 0: print("################################################\n")
if verbose > 0: print("Converting merged data to long format")
if verbose > 0: toc()
if verbose > 0: print("################################################\n")

# Extract the desired quantity from the merged data :
def standard_score_max(col):
  return np.max(( col - np.mean(col) ) / np.std(col))

df_std = pd.DataFrame(df.filter(regex='Value').apply(standard_score_max, raw=False,

# Function to split the ImageJ file name
def split(delimiters, string, maxsplit=0):
  import re
  regexPattern = '|'.join(map(re.escape, delimiters))
  return re.split(regexPattern, str(string), maxsplit)

# The "_" is my choice, the "/" is made by LASX when it finds the names of your regi
delimiters = "_", "/"

# Converts the index of the df to type "variable:value,variable:value,..."
def isolate_variables (index):
  variables = ['f', 'm', 'ZT6', 'ZT18', 'TPer2KO', 'Per2flfl', 'DorCX', 'Thal', 'Hyp
  names = ['sex', 'genotype', 'timepoint', 'region']
  result = []
  for x in split(delimiters, index) :
    if x in variables:
      result.append(''.join(filter(str.isalnum, x)))
  return ','.join(a + ':' + b for a,b in zip(names, result))

# Makes a column with the variable:value,variable:value,... format
df_std['variables'] = df_std.index.map(isolate_variables)

# Makes each variable a column and attributes the correct value to each row
df_std_long = df_std['variables'].str.split(",", expand=True)
cols = []
for col in df_std_long:
  df_temp = df_std_long[col].str.split(":", expand=True)
  df_std_long[col] = df_temp[1]
  cols.append(df_temp.iloc[0, 0])
df_std_long.columns = cols
df_std_long['intensity'] = df_std[0]

if write_excel :
  if verbose > 0: print("\tSaving data to " + path +'Long_data.xlsx')
  df_std_long.to_excel(path+'Long_data.xlsx')

# Make df for report :
df_std_long_counts = df_std_long.value_counts(subset=['sex', 'genotype', 'timepoint'
df_std_long_counts.loc["Total"] = df_std_long_counts.sum()

df_std_long_counts_region = df_std_long.value_counts(subset=['sex', 'genotype', 'tim
df_std_long_counts_region.loc["Total"] = df_std_long_counts_region.sum()

df_std_long_counts_Only_region = df_std_long.value_counts(subset=['region'], sort=Fa
df_std_long_counts_Only_region.loc["Total"] = df_std_long_counts_Only_region.sum()
```

```
379
380   if write_excel :
381     if verbose > 0: print("\tSaving data to " + path +'Long_data.xlsx')
382     df_std_long_counts.to_excel(path+'Counts.xlsx')
383     df_std_long_counts_region.to_excel(path+'Counts_regions.xlsx')
384     df_std_long_counts_Only_region.to_excel(path+'Counts_only_regions.xlsx')
385
386   if tex :
387     df_std_long_counts.to_latex(buf=path+'Counts.tex', header=True , index=True , na_r
388     df_std_long_counts_region.to_latex(buf=path+'Counts_regions.tex', header=True , in
389     df_std_long_counts_Only_region.to_latex(buf=path+'Counts_only_regions.tex', header
390
391   return df_std_long
392
393 def make_plot(df, sex, genotype, regions, verbose, path, write_excel):
394   """
395     Fct that plots and saves a figure for each region.
396     Each plot as day and night
397   """
398   # pip install SciencePlots
399   import pandas as pd
400   import matplotlib.pyplot as plt
401   import numpy as np
402
403   if verbose > 0: tic()
404   if verbose > 0: print("################################################\n")
405   if verbose > 0: print("Making plots :")
406
407   # New directory to save the Boxplot data and plots
408   new_dir = path+genotype+'\\'
409   try:
410     os.makedirs(new_dir)
411   except OSError:
412     if verbose > 0: print ('Failed to make directory :', new_dir, ' to store the data.
413   else:
414     if verbose > 0: print('Made directory :', new_dir, ' to store the data sorted by g
415
416   path = path+genotype+'\\'
417
418   # Filter by genotype
419   if verbose > 0: print("\tFiltering by genotype :", genotype)
420   df = df.filter(regex=f'(?=.*{genotype})|dist').dropna(axis=0, how='all')
421
422   #DONE need to do the normalisation step here, not after the mean calculations.
423   # Subtract the offset and devide by the variance of the background ?
424
425   def standard_score(col):
426     """
427      The standard score is the number of standard deviations by which the value of a r
428      (i.e., an observed value or data point) is above or below the mean value
429      of what is being observed or measured.
430      It is calculated by subtracting the population mean from an individual raw score
431      and then dividing the difference by the population standard deviation.
432      https://en.wikipedia.org/wiki/Standard_score
433     """
434     return ( col - np.mean(col) ) / np.std(col)
435
436   # Make df with the standard score
437   df_standard_score = pd.DataFrame(df.filter(regex='Value').apply(standard_score, raw=
438   df_standard_score['dist'] = df['dist']
```

```
439
440   # Add the mean and sem for each region and time
441   if verbose > 0: print("\tComputing mean and SEM values for :")
442   for region in regions:
443     if verbose > 0: print("\t\t", region, 'night')
444     df[region,'Mean_ZT18'] = df.filter(regex=f'(?=.*{region})(?=.*ZT18)(?=.*Value)').m
445     df[region,'SEM_ZT18'] = df.filter(regex=f'(?=.*{region})(?=.*ZT18)(?=.*Value)').se
446
447     if verbose > 0: print("\t\tStandard score : ", region, 'night')
448     df_standard_score[region,'Mean_ZT18'] = df_standard_score.filter(regex=f'(?=.*{reg
449     df_standard_score[region,'SEM_ZT18'] = df_standard_score.filter(regex=f'(?=.*{regi
450
451     if verbose > 0: print("\t\t", region, 'day')
452     df[region,'Mean_ZT6'] = df.filter(regex=f'(?=.*{region})(?=.*ZT6)(?=.*Value)').mea
453     df[region,'SEM_ZT6'] = df.filter(regex=f'(?=.*{region})(?=.*ZT6)(?=.*Value)').sem(
454
455     if verbose > 0: print("\t\tStandard score : ", region, 'night')
456     df_standard_score[region,'Mean_ZT6'] = df_standard_score.filter(regex=f'(?=.*{regi
457     df_standard_score[region,'SEM_ZT6'] = df_standard_score.filter(regex=f'(?=.*{regio
458
459   # Save the Datframe to .xlsx for reference
460   if verbose and write_excel > 0: print("\n\tWriting DataFrame to :", path+'Filtered_'
461   if write_excel :
462     df.to_excel(path+'Filtered_'+genotype+'_data.xlsx')
463     df_standard_score.to_excel(path+'Filtered_standard_score_'+genotype+'_data.xlsx')
464
465   # Make the plots
466   if verbose > 0: print("\n\tPloting data\n")
467
468   # Figure parameters
469   n_rows = len(regions)
470   n_cols = 1
471   figsize = (5, len(regions) * 5 + 2)
472   title_font_size = 20
473   plt.style.use(['science','grid'])
474   plt.tight_layout(pad=0.5)
475
476   # Line plots
477
478   fig_line, axes_line = plt.subplots(nrows=n_rows, ncols=n_cols, figsize=figsize)
479   fig_line.subplots_adjust(top=0.90)
480   fig_line.suptitle('Average intensity of AQP-4 staining for ' + genotype + ' mice.',
481
482   for ax, region in zip(axes_line.flatten(), regions):
483     ax.grid()
484     ax.plot(df['dist'], df[region,'Mean_ZT18'], 'r', linestyle='', marker='.', label='
485     ax.fill_between(df['dist'], df[region,'Mean_ZT18'] - df[region,'SEM_ZT18'], df[reg
486     ax.plot(df['dist'], df[region,'Mean_ZT6'], 'g', linestyle='', marker='.', label='Z
487     ax.fill_between(df['dist'], df[region,'Mean_ZT6'] - df[region,'SEM_ZT6'], df[regio
488     ax.set(title=region, xlabel='Distance ($\mu$m)', ylabel=('Intensity (A.U)'))
489     ax.grid()
490     ax.legend()
491
492   plt.savefig(path + genotype + '_mice_' + 'LinePlot.pdf')
493   plt.close()
494
495   if verbose > 0: print('\tSaved Line plot as', path + genotype + '_mice_' + 'LinePlot
496
497   # Line Standard score plot
498
```

```
fig_standard_score, axes_standard_score = plt.subplots(nrows=n_rows, ncols=n_cols, f
# fig_standard_score.subplots_adjust(hspace=0.1)
# fig_standard_score.suptitle('Average intensity of AQP-4 staining for ' + genotype

for ax, region in zip(axes_standard_score.flatten(), regions):
  ax.grid()
  ax.plot(df_standard_score['dist'], df_standard_score[region,'Mean_ZT18'], 'r', lin
  ax.fill_between(df_standard_score['dist'], df_standard_score[region,'Mean_ZT18'] -
  ax.plot(df_standard_score['dist'], df_standard_score[region,'Mean_ZT6'], 'g', line
  ax.fill_between(df_standard_score['dist'], df_standard_score[region,'Mean_ZT6'] -
  ax.set(title=region, xlabel='Distance ($\mu$m)', ylabel=('Intensity (STD)'))
  ax.grid()
  ax.legend()

plt.savefig(path + genotype + '_mice_' + 'LinePlot_standard_score.pdf')
plt.close()

if verbose > 0: print('\tSaved Line plot as', path + genotype + '_mice_' + 'LinePlot

# Line plots minus the background

# Function that subtracts the mean of the 1st and last 3rd from the array
def substract_background(col):
  return col - np.mean(col[list(list(range(0,round(len(col)*1/3))) + list(range(roun

df_Back_Sub = pd.DataFrame(df.filter(regex='Mean|SEM').apply(substract_background, r
df_Back_Sub['dist'] = df['dist']

fig_line_backsub, axes_line_backsub = plt.subplots(nrows=n_rows, ncols=n_cols, figsi
fig_line_backsub.subplots_adjust(top=0.90)
fig_line_backsub.suptitle('Average intensity of AQP-4 staining for ' + genotype + '

for ax, region in zip(axes_line_backsub.flatten(), regions):
  ax.grid()
  ax.plot(df_Back_Sub['dist'], df_Back_Sub[region,'Mean_ZT18'], 'r', linestyle='', m
  ax.fill_between(df_Back_Sub['dist'], df_Back_Sub[region,'Mean_ZT18'] - df_Back_Sub
  ax.plot(df_Back_Sub['dist'], df_Back_Sub[region,'Mean_ZT6'], 'g', linestyle='', ma
  ax.fill_between(df_Back_Sub['dist'], df_Back_Sub[region,'Mean_ZT6'] - df_Back_Sub[
  ax.set(title=region, xlabel='Distance ($\mu$m)', ylabel=('Intensity (A.U)'))
  ax.grid()
  ax.legend()

plt.savefig(path + genotype + '_mice_' + '_BackGroundSub_LinePlot.pdf')
plt.close()

if verbose > 0: print('\tSaved Line plot with background subtracted as', path + geno

# Boxplots
# DONE Each dot in the box plot is an individual blood vessel, not the average of al
# DONE Add statistical analysis

# Function that subtracts the mean of the 1st and last 3rd from the array and return
def substract_background_max(col):
  return np.max(col - np.mean(col[list(list(range(0,round(len(col)*1/3))) + list(ran

def standard_score_max(col):
  return np.max(( col - np.mean(col) ) / np.std(col))

# Retrives the values for the boxplot
```

```
559  df_values = pd.DataFrame(df.filter(regex='Value').apply(substract_background_max, ra
560  df_values_standard_score = pd.DataFrame(df.filter(regex='Value').apply(standard_scor
561
562  # Make a multiindexed df for the boxplot
563  iterables = [regions, ['ZT6', 'ZT18']]
564  index = pd.MultiIndex.from_product(iterables, names=["region", "time"])
565
566  # Fills the df with the correct data for each region and time point
567  df_box = pd.DataFrame(index=index).T
568  for region in regions:
569    df_box[region, 'ZT6'] = pd.Series(df_values.filter(regex=f'(?=.*ZT6)(?=.*{region})
570    df_box[region, 'ZT18'] = pd.Series(df_values.filter(regex=f'(?=.*ZT18)(?=.*{region
571
572  df_box_standard_score = pd.DataFrame(index=index).T
573  for region in regions:
574    df_box_standard_score[region, 'ZT6'] = pd.Series(df_values_standard_score.filter(r
575    df_box_standard_score[region, 'ZT18'] = pd.Series(df_values_standard_score.filter(
576
577
578  def min_max_rescale(x):
579    #TODO make this rescale with the extrema of the entire group, not inside each one.
580    return (x-x.min())/(x.max()-x.min())
581
582  # Apply min-max normalization. Gives values between 0 and 1
583  # df_box = df_box.apply(min_max_rescale, axis=0)
584
585  # BoxPlots background sub
586  fig_Box, axes_Box = plt.subplots(nrows=n_rows, ncols=n_cols, figsize=figsize)
587  fig_Box.subplots_adjust(top=0.90)
588  fig_Box.suptitle('Average intensity of AQP-4 staining for ' + genotype + ' mice.\nBa
589
590  # Dataframe for storing statistical data
591  df_box_stats = pd.DataFrame(data=None, index=['statistic', 'pvalue'], columns=region
592
593  for ax, region in zip(axes_Box.flatten(), regions):
594    df_box[region].boxplot(ax=ax, )
595    ax.set(title=region, xlabel='Time point', ylabel=('Intensity (A.U)'))
596    #ax.legend()
597
598    # Statistical analysis
599    # TODO find out why the p-value changes when running twice on the same data
600    from scipy import stats
601
602    if verbose > 0: print('\tPerforming statistical analysis on the region : ' + regio
603
604    region_stats = stats.ttest_ind(df_box[region, 'ZT6'].dropna(), df_box[region, 'ZT1
605    df_box_stats.loc['statistic', region] = region_stats.statistic
606    df_box_stats.loc['pvalue', region] = region_stats.pvalue
607    if verbose > 0: print(region_stats) #TODO Make sure the stats and pvalue are the r
608
609    # Annotate the boxplots
610    height = np.max(df_box[region].max(axis=1))*np.ones(len(regions))
611    bars = np.arange(len(regions))
612    barplot_annotate_brackets(1, 2, region_stats.pvalue, bars, height, ax)
613
614  plt.savefig(path + genotype + '_mice_' + 'Box_Plot_background_subed.pdf')
615  plt.close()
616
617  if verbose > 0: print('\tSaved Box plot with background subtracted as', path + genot
618
```

```
619   if write_excel : df_box.to_excel(path + 'Boxplot_' + genotype + '_data.xlsx')
620   if write_excel : df_box_stats.to_excel(path + 'df_box_stats_background_sub' + genoty
621
622   # BoxPlots Standard score
623   fig_Box_standard_score, axes_Box_standard_score = plt.subplots(nrows=n_rows, ncols=n
624   fig_Box_standard_score.subplots_adjust(top=0.90)
625   fig_Box_standard_score.suptitle('Average intensity of AQP-4 staining for ' + genotyp
626
627   # Dataframe for storing statistical data
628   df_box_stats_std = pd.DataFrame(data=None, index=['statistic', 'pvalue'], columns=re
629
630   for ax, region in zip(axes_Box_standard_score.flatten(), regions):
631     df_box_standard_score[region].boxplot(ax=ax, )
632     ax.set(title=region, xlabel='Time point', ylabel=('Intensity (STD)'))
633     #ax.legend()
634
635     # Statistical analysis
636     # TODO find out why the p-value changes when running twice on the same data
637     from scipy import stats
638
639     if verbose > 0: print('\tPerforming statistical analysis on the region : ' + regio
640
641     region_stats = stats.ttest_ind(df_box_standard_score[region, 'ZT6'].dropna(), df_b
642     df_box_stats_std.loc['statistic', region] = region_stats.statistic
643     df_box_stats_std.loc['pvalue', region] = region_stats.pvalue
644     if verbose > 0: print(region_stats) #TODO Make sure the stats and pvalue are the r
645
646     # Annotate the boxplots
647     height = np.max(df_box_standard_score[region].max(axis=1))*np.ones(len(regions))
648     bars = np.arange(len(regions))
649     barplot_annotate_brackets(1, 2, region_stats.pvalue, bars, height, ax)
650
651   plt.savefig(path + genotype + '_mice_' + 'Box_Plot_standard_score.pdf')
652   plt.close()
653
654   if verbose > 0: print('\n\tSaved Box plot with standard scores as', path + genotype
655
656   if write_excel : df_box_standard_score.to_excel(path + 'Boxplot_standard_score_' + g
657   if write_excel : df_box_stats_std.to_excel(path + 'df_box_stats_std_' + genotype + '
658
659   if verbose > 0: toc()
660   if verbose > 0: print("#################################################\n")
661
662   return(0)
663
664 def make_boxplots(df, sex, regions, verbose, path, write_excel, tex):
665   """
666     Fct that plots and saves a figure for each region.
667     Each plot as day and night or genotype
668   """
669   # pip install SciencePlots
670   import pandas as pd
671   import matplotlib.pyplot as plt
672   import numpy as np
673   #import pingouin as pg
674
675   if verbose > 0: tic()
676   if verbose > 0: print("#################################################\n")
677   if verbose > 0: print("Making box plots for ANOVA :")
678
```

```
679  # New directory to save the Boxplot data and plots
680  new_dir = path+'ANOVA'+'\\'
681  try:
682    os.makedirs(new_dir)
683  except OSError:
684    if verbose > 0: print ('Failed to make directory :', new_dir, ' to store the Boxpl
685  else:
686    if verbose > 0: print('Made directory :', new_dir, ' to store the BoxPlots\n')
687
688  path_box = path+'ANOVA'+'\\'
689
690  # Genotypes to use with the regex filters
691  genotypes = {'KO':'TPer2KO', 'CO':'Per2flfl'}
692
693  # Filter by genotype
694  if verbose > 0: print("\tFiltering by genotype and or time point :")
695
696  # Separate either by time point (ZT6/ZT18) or by genotype (KO/CO)
697  df_CO = df.filter(regex=f'(?=.*{genotypes["CO"]})|dist').dropna(axis=0, how='all')
698  df_KO = df.filter(regex=f'(?=.*{genotypes["KO"]})|dist').dropna(axis=0, how='all')
699  df_ZT6 = df.filter(regex=f'(?=.*ZT6)|dist').dropna(axis=0, how='all')
700  df_ZT18 = df.filter(regex=f'(?=.*ZT18)|dist').dropna(axis=0, how='all')
701
702  # Save Datframes to .xlsx for reference
703  if verbose and write_excel > 0: print("\n\tWriting DataFrame to :", path_box+'DataFr
704  if write_excel :
705    df_CO.to_excel(path_box+'DataFrame_BoxPlots_CO_data.xlsx')
706    df_KO.to_excel(path_box+'DataFrame_BoxPlots_KO_data.xlsx')
707    df_ZT6.to_excel(path_box+'DataFrame_BoxPlots_ZT6_data.xlsx')
708    df_ZT18.to_excel(path_box+'DataFrame_BoxPlots_ZT18_data.xlsx')
709
710  def standard_score_max(col):
711    return np.max(( col - np.mean(col) ) / np.std(col))
712
713  # Retrives the values for the boxplot
714  df_CO_std = pd.DataFrame(df_CO.filter(regex='Value').apply(standard_score_max, raw=F
715  df_KO_std = pd.DataFrame(df_KO.filter(regex='Value').apply(standard_score_max, raw=F
716  df_ZT6_std = pd.DataFrame(df_ZT6.filter(regex='Value').apply(standard_score_max, raw
717  df_ZT18_std = pd.DataFrame(df_ZT18.filter(regex='Value').apply(standard_score_max, r
718
719  # Make 2 multiindexed dfs for the boxplots
720  iterables_time_points = [regions, ['ZT6', 'ZT18']]
721  index_time_points = pd.MultiIndex.from_product(iterables_time_points, names=["region
722
723  iterables_genotypes = [regions, [genotypes['CO'], genotypes['KO']]]
724  index_genotypes = pd.MultiIndex.from_product(iterables_genotypes, names=["region", "
725
726  # Fills the dfs with the correct data for each region and time point
727  df_box_CO = pd.DataFrame(index=index_time_points).T
728  for region in regions:
729    df_box_CO[region, 'ZT6'] = pd.Series(df_CO_std.filter(regex=f'(?=.*ZT6)(?=.*{regio
730    df_box_CO[region, 'ZT18'] = pd.Series(df_CO_std.filter(regex=f'(?=.*ZT18)(?=.*{reg
731
732  df_box_KO = pd.DataFrame(index=index_time_points).T
733  for region in regions:
734    df_box_KO[region, 'ZT6'] = pd.Series(df_KO_std.filter(regex=f'(?=.*ZT6)(?=.*{regio
735    df_box_KO[region, 'ZT18'] = pd.Series(df_KO_std.filter(regex=f'(?=.*ZT18)(?=.*{reg
736
737  df_box_ZT6 = pd.DataFrame(index=index_genotypes).T
738  for region in regions:
```

```
    df_box_ZT6[region, genotypes['CO']] = pd.Series(df_ZT6_std.filter(regex=f'(?=.*{ge
    df_box_ZT6[region, genotypes['KO']] = pd.Series(df_ZT6_std.filter(regex=f'(?=.*{ge

  df_box_ZT18 = pd.DataFrame(index=index_genotypes).T
  for region in regions:
    df_box_ZT18[region, genotypes['CO']] = pd.Series(df_ZT18_std.filter(regex=f'(?=.*{
    df_box_ZT18[region, genotypes['KO']] = pd.Series(df_ZT18_std.filter(regex=f'(?=.*{

  # Master DataFrame for boxplots
  df_box_ZT6.columns = pd.MultiIndex.from_product(df_box_ZT6.columns.levels + [['ZT6']
  df_box_ZT18.columns = pd.MultiIndex.from_product(df_box_ZT18.columns.levels + [['ZT1
  df_box_master = pd.concat([df_box_ZT6, df_box_ZT18], axis=1).swaplevel('timepoint',

  # Save Datframe to .xlsx for reference
  if verbose and write_excel > 0: print("\n\tWriting Master Box DataFrame to :", path_
  if write_excel :
    df_box_master.to_excel(path_box+'DataFrame_BoxPlots_MASTER.xlsx')

  # Make the BoxPlots

  # New directory to save the Boxplot stats
  new_dir_stats = path_box+'Stats'+'\\'
  try:
    os.makedirs(new_dir_stats)
  except OSError:
    if verbose > 0: print ('Failed to make directory :', new_dir_stats, ' to store the
  else:
    if verbose > 0: print('Made directory :', new_dir_stats, ' to store the BoxPlots s

  path_box_stats = path_box+'Stats'+'\\'

  # Figure parameters
  n_rows = 2
  n_cols = 2
  figsize = (12,12)
  title_font_size = 20
  plt.style.use(['science','grid'])
  plt.tight_layout(pad=0.5)

  # One subplot per region. Each subplot with 4 boxes ZT6 (CO+KO) and ZT18 (CO+KO)
  fig_Box, axes_Box = plt.subplots(nrows=n_rows, ncols=n_cols, figsize=figsize)
  fig_Box.subplots_adjust(top=0.90)
  fig_Box.suptitle('Maximum intensity of AQP-4 staining\nStandard Score.', fontsize=ti

  # Old t-test
  #Dataframe for storing statistical data
  #df_box_stats = pd.DataFrame(data=None, index=['statistic', 'pvalue'], columns=regio

  # List to store the dfs created in the loop
  list_df_box_var_timepoint = list()
  list_df_box_var_genotype = list()
  list_df_box_normal = list()

  list_df_box_anova_stats = list()
  list_df_box_anova_data = list()

  for ax, region in zip(axes_Box.flatten(), regions):
    df_box_master[region].boxplot(ax=ax) #TODO is this the correct data ? check with t
    ax.set(title=region, xlabel='Time point and genotype', ylabel=('Intensity (STD)'))

```

```
    # Statistical analysis
    if verbose > 0: print('\n\tPerforming statistical analysis on the region : ' + reg

    import pingouin as pg

    #TODO Test if the groups are normally distributed.
    # pingouin.homoscedasticity(data, dv=None, group=None, method='levene', alpha=0.05
    # pg.normality(data, method='normaltest', alpha=0.05).round(3)
    # when the groups have unequal variances, it is best to use the Welch ANOVA (pingo

    # Reshape the dataframe to remove the multi level column names
    df_box_anova = df_box_master[region].stack(level=[0,1]).reset_index(level=['timepo
    list_df_box_anova_data.append(df_box_anova)

    # Test for equal variance
    df_box_var_timepoint = pg.homoscedasticity(df_box_anova, dv='intensity', group='ti
    df_box_var_genotype = pg.homoscedasticity(df_box_anova, dv='intensity', group='gen
    if verbose > 0: print('\t\tTimepoint :\n', df_box_var_timepoint, '\n')
    if verbose > 0: print('\t\tGenotype :\n', df_box_var_genotype)

    list_df_box_var_timepoint.append(df_box_var_timepoint)
    list_df_box_var_genotype.append(df_box_var_genotype)

    # Test for normality
    if df_box_master[region].count().min() > 7:
      df_box_normal = pg.normality(df_box_master[region], method='normaltest', alpha=0
      if verbose > 0: print('\t\tNormality test :\n', df_box_normal)
      list_df_box_normal.append(df_box_normal)

    # Print warining if the data does not have equal varinance or if not normal
    if not df_box_var_timepoint['equal_var'].all():
      if verbose > 0: print('\n\t\t#####################################################
      #ax.set(title=region + '\nANOVA is not accurate :\ndata between timepoints does

    if not df_box_var_genotype['equal_var'].all():
      if verbose > 0: print('\n\t\t#####################################################

    if not df_box_normal['normal'].all():
      if verbose > 0: print('\n\t\t#####################################################
      ax.set(title=region + '\nANOVA not accurate, data not normally distributed', xla

    # Perform the two-way ANOVA
    #TODO Is a Welch ANOVA better ?
    df_box_anova_stats = df_box_anova.anova(dv='intensity', between=['timepoint', 'gen
    list_df_box_anova_stats.append(df_box_anova_stats)

    # Annotate the boxplots
    #height = np.max(df_box_master[region].max(axis=1))*np.ones(len(regions))
    #bars = np.arange(len(regions))
    #barplot_annotate_brackets(1, 2, region_stats.pvalue, bars, height, ax)

    # OLD t-test
    # region_stats = stats.ttest_ind(df_box_ZT6[region, genotypes['CO']].dropna(), df_
    # df_box_stats.loc['statistic', region] = region_stats.statistic
    # df_box_stats.loc['pvalue', region] = region_stats.pvalue
    # print(region_stats) #TODO Make sure the stats and pvalue are the right way aroun


  plt.savefig(path_box + 'df_box_master.pdf')
  plt.close()
```

```
859
860   if verbose > 0: print('\nSaved Box plot with standard scores as', path_box + 'df_box
861
862   # Concat the data from the equal variance tests :
863   df_box_var_timepoint = pd.concat(list_df_box_var_timepoint, keys=regions, axis=1)
864   df_box_var_genotype = pd.concat(list_df_box_var_genotype, keys=regions, axis=1)
865   df_box_var = pd.concat([df_box_var_timepoint,df_box_var_genotype], keys=['timepoint'
866
867   # Concat the data from the normailty tests :
868   df_box_normal = pd.concat(list_df_box_normal, keys=regions)
869
870   # Concat the data from the ANOVA tests :
871   df_box_anova_data = pd.concat(list_df_box_anova_data, keys=regions, axis=0)
872   df_box_anova_stats = pd.concat(list_df_box_anova_stats, keys=regions)
873
874   # Save the data
875   if verbose > 0: print('\nSaving Box plot data and stats to', path_box_stats, '\n')
876   if write_excel : df_box_var.to_excel(path_box_stats + 'df_box_var_stats.xlsx')
877   if write_excel : df_box_normal.to_excel(path_box_stats + 'df_box_normal_stats.xlsx')
878   if write_excel : df_box_anova_stats.to_excel(path_box_stats + 'df_box_anova_stats.xl
879   if write_excel : df_box_anova_data.to_excel(path_box_stats + 'df_box_anova_data.xlsx
880
881   if tex : df_box_var.to_latex(buf=path_box_stats+'df_box_var.tex', header=True , inde
882   if tex : df_box_normal.to_latex(buf=path_box_stats+'df_box_normal.tex', header=True
883   if tex : df_box_anova_stats.to_latex(buf=path_box_stats+'df_box_anova_stats.tex', he
884   if tex : df_box_anova_data.to_latex(buf=path_box_stats+'df_box_anova_data.tex', head
885
886
887   if verbose > 0: toc()
888   if verbose > 0: print("#################################################\n")
889
890   # Print the results
891   if verbose > 0: print('Results from the two-way ANOVA on the data :\n\n', df_box_ano
892   if verbose > 0: print("\n'Source': Factor names\n'SS': Sums of squares\n'DF': Degree
893   if verbose > 0: print("\n#################################################\n")
894
895   return(0)
896
897 def make_boxplots_long (df, regions, verbose, path, write_excel, tex):
898   '''
899   Takes the dataframe output from make_std_long and makes
900   a boxplot comparing the sexs at each region in regions
901   Uses the function barplot_annotate_brackets
902   It makes plottin easier :
903   df.groupby('region').boxplot(column='intensity', by=['genotype', 'sex'], figsize=(15
904   but you have no control to annotate each subplot seperatly since there is no loop
905   '''
906
907   # pip install SciencePlots
908   import pandas as pd
909   import matplotlib.pyplot as plt
910   import numpy as np
911   from scipy import stats
912   import seaborn as sns
913
914   if verbose > 0: tic()
915   if verbose > 0: print("#################################################\n")
916   if verbose > 0: print("Making box plots to compare sexes :")
917
918   # Figure parameters
```

```
n_rows = 2
n_cols = 2
figsize = (10,8)
title_font_size = 20
plt.style.use(['science','grid'])
plt.tight_layout(pad=0.5)

# New directory to save the Boxplot data and plots
new_dir = path+'Long'+'\\'
try:
  os.makedirs(new_dir)
except OSError:
  if verbose > 0: print ('Failed to make directory :', new_dir, ' to store the Boxpl
else:
  if verbose > 0: print('Made directory :', new_dir, ' to store the BoxPlots\n')

path_box = path+'Long'+'\\'

# Make violin plots with all the data
fig_violin, axes_violin = plt.subplots(nrows=1, ncols=1, figsize=(6,6))
# fig_violin.subplots_adjust(hspace=1)
fig_violin.suptitle('Maximum intensity of AQP-4 staining\nStandard Score.', fontsize

axes_violin = sns.violinplot(data=df, x='genotype', hue='timepoint', y='intensity',

plt.savefig(path_box + 'ViolinPlot.pdf')
plt.close()

# # This unfortunatly doesnt work "groupby"
# # Violin separated by region
# fig_violin_region, axes_violin_region = plt.subplots(nrows=1, ncols=1, figsize=(6,
# # fig_violin.subplots_adjust(hspace=1)
# fig_violin.suptitle('Maximum intensity of AQP-4 staining\nStandard Score.', fontsi

# axes_violin_region = sns.violinplot(data=df.groupby('region'), x='genotype', hue='

# plt.savefig(path_box + 'ViolinPlot_regions.pdf')
# plt.close()

# # Set the region column as an index to be able to select data by region
# df.set_index(keys='region', append=True,  inplace=True)

# # One subplot comparing sex per region
# fig_Box_sex, axes_Box_sex = plt.subplots(nrows=n_rows, ncols=n_cols, figsize=figsi
# fig_Box_sex.subplots_adjust(top=0.90)
# fig_Box_sex.suptitle('Maximum intensity of AQP-4 staining\nStandard Score.', fonts

# # Old t-test
# #Dataframe for storing statistical data
# df_box_sex_stats = pd.DataFrame(data=None, index=['statistic', 'pvalue'], columns=

# for ax, region in zip(axes_Box_sex.flatten(), regions):
#   df.T.swaplevel(axis=1)[region].T.boxplot(column=['intensity'], by=['genotype', '
#   ax.set(title=region, xlabel='Sex', ylabel=('Intensity (STD)'))

#   # Statistical analysis
#   if verbose > 0: print('\n\tPerforming statistical analysis on the region : ' + r

#   # t-test
#   # Get the correct columns
```

```
#    df_m = df.T.swaplevel(axis=1)[region].T.set_index('sex', append=True).T.swapleve
#    df_f = df.T.swaplevel(axis=1)[region].T.set_index('sex', append=True).T.swapleve

#    from scipy import stats
#    region_stats = stats.ttest_ind(df_m.dropna(), df_f.dropna(), axis=0, equal_var=F
#    df_box_sex_stats.loc['statistic', region] = region_stats.statistic
#    df_box_sex_stats.loc['pvalue', region] = region_stats.pvalue
#    if verbose : print('\t',region_stats, '\n') #TODO Make sure the stats and pvalue

#    # Annotate the boxplots
#    height = np.max(df.T.swaplevel(axis=1)[region].T['intensity'].max())*np.ones(len
#    bars = np.arange(len(regions))
#    barplot_annotate_brackets(1, 2, region_stats.pvalue, bars, height, ax)

# fig_Box_sex.suptitle('Maximum intensity of AQP-4 staining\nStandard Score.', fonts
# plt.savefig(path_box + 'Box_Plot_Sex.pdf')
# plt.close()

sex_dir = path_box+'Sex'+'\\'
try:
  os.makedirs(sex_dir)
except OSError:
  if verbose > 0: print ('Failed to make directory :', sex_dir, ' to store the Boxpl
else:
  if verbose > 0: print('Made directory :', sex_dir, ' to store the BoxPlots\n')

path_sex = sex_dir

#Dataframe for storing statistical data
df_box_sex_stats = pd.DataFrame(data=None, index=['statistic', 'pvalue'], columns=re

for region in regions:
  # Iterate through the regions
  df_sex = df[df['region'] == region]

  # Group the data to your liking
  grouped = df_sex.groupby('genotype')

  # Figure out number of rows needed for 2 column grid plot
  # Also accounts for odd number of plots
  import math
  nrows = int(math.ceil(len(grouped)/2.))

  #Setup Subplots
  fig, axs = plt.subplots(nrows,2, figsize=(10,5))

  for (name, df_sex), ax in zip(grouped, axs.flat):
    # t-test
    # Get the correct columns
    df_m = df_sex.set_index('sex', append=True).T.swaplevel(axis=1)['m'].T['intensit
    df_f = df_sex.set_index('sex', append=True).T.swaplevel(axis=1)['f'].T['intensit

    from scipy import stats
    region_stats = stats.ttest_ind(df_m.dropna(), df_f.dropna(), axis=0, equal_var=F
    df_box_sex_stats.loc['statistic', region] = region_stats.statistic
    df_box_sex_stats.loc['pvalue', region] = region_stats.pvalue

    # Plot
    df_sex.boxplot(column=['intensity'], by=['genotype', 'sex'], ax=ax)

```

```
      # Annotate the boxplots
      height = np.max(df['intensity'].max())*np.ones(len(regions))
      bars = np.arange(len(regions))
      barplot_annotate_brackets(1, 2, region_stats.pvalue, bars, height, ax)

      # Subplot title
      ax.set_title(region)
      ax.set_ylabel('Intensity (STD)')

    plt.savefig(path_sex+'Sex_'+region+'.pdf')

  df_box_sex_stats


  if verbose > 0: print('\n\tSaved Box plot with standard scores as', path_sex+'Sex_'+

  if tex : df_box_sex_stats.to_latex(buf=path_sex+'df_box_sex_stats.tex', header=True

  if write_excel : df_box_sex_stats.to_excel(path_sex + 'df_box_sex_stats_data.xlsx')

  # # One subplot per region with violin plots this time
  # fig_violin_region, axes_violin_region = plt.subplots(nrows=n_rows, ncols=n_cols, f
  # fig_violin_region.subplots_adjust(top=0.90)
  # fig_violin_region.suptitle('Maximum intensity of AQP-4 staining\nStandard Score.',

  # for ax, region in zip(axes_violin_region.flatten(), regions):
  #   ax = sns.violinplot(data=df.T.swaplevel(axis=1)[region].T, x='genotype', hue='ti
  #   # ax.set(title=region, xlabel='Sex', ylabel=('Intensity (STD)'))

  # fig_violin_region.suptitle('Maximum intensity of AQP-4 staining\nStandard Score.',
  # plt.savefig(path_box + 'Violin_regions.pdf')
  # plt.close()

  if verbose > 0: toc()
  if verbose > 0: print("#################################################\n")

  return 0

############################################################

def main():
  """Make plots from excel files in directory."""

  # Parse arguments
  args = parseArgs()

  # Convert object elements to standard variables for functions
  path = args.path + '\\' # Useful when coping path from win explorer
  regions = args.regions
  verbose = args.verbose
  genotypes = args.genotypes
  sex = args.sex
  write_excel = args.excel
  tex = args.tex

  if verbose > 0: tic()
  if verbose > 0: print("\n\n#################################################\n")

  # Make directory to save results
  new_dir = path+'Data_and_Plots'+'\\'
```

```
  try:
    os.makedirs(new_dir)
  except OSError:
    if verbose > 0: print ('Failed to make directory :', new_dir, ' to store the resul
  else:
    if verbose > 0: print('Made directory :', new_dir, ' to store the results\n')

  # Start calling functions to do the heavy lifting

  df_raw = make_big_DataFrame(path, verbose)
  if verbose > 0 and write_excel > 0: print('Saving raw data\n')
  if write_excel : df_raw.to_excel(new_dir+'Raw_data.xlsx')

  # Merge with bins
  df_merged_bins = merge_on_distance(df_raw, verbose, bins=True)
  if verbose and write_excel > 0: print('Saving merged data\n')
  if write_excel : df_merged_bins.to_excel(new_dir+'Merged_bins.xlsx')

  # Merge without bins
  df_merged_no_bins = merge_on_distance(df_raw, verbose, bins=False)
  if verbose and write_excel > 0: print('Saving merged data\n')
  if write_excel : df_merged_no_bins.to_excel(new_dir+'Merged_no_bins.xlsx')

  df_std_long = make_std_long(df_merged_no_bins, verbose, new_dir, write_excel, tex)

  for genotype in genotypes:
    if verbose : print('\n\n\n##############################',genotype)
    make_plot(df_merged_bins, sex, genotype, regions, verbose, new_dir, write_excel)
  #make_plot(df_merged, sex, genotypes, regions, verbose, new_dir, write_excel)

  # New directory to save the Boxplot data and plots
  box_dir = new_dir+'BoxPlots'+'\\'
  try:
    os.makedirs(new_dir)
  except OSError:
    if verbose > 0: print ('Failed to make directory :', box_dir, ' to store the Boxpl
  else:
    if verbose > 0: print('Made directory :', box_dir, ' to store the BoxPlots\n')

  make_boxplots(df_merged_no_bins, sex, regions, verbose, box_dir, write_excel, tex)

  make_boxplots_long(df_std_long, regions, verbose, box_dir, write_excel, tex)


if __name__ == '__main__':
  import time
  start_time = time.time()
  main()
  print("\nTotal running time: {:.3f} seconds.".format(time.time() - start_time))
```

Listing 2: Python code written for data processing — stastitical analysis and graphing.

8.6 Figures

List of Figures

8.7 Tables

List of Tables

9 References

References

[1] abcam. Anti-gfap antibody (ab53554). Online, 2021. https://www.abcam.com/gfap-antibody-ab53554.html visited on 2021-08-31.

[2] abcam. Donkey anti-rabbit igg (ab175693). Online, 2021. https://www.abcam.com/donkey-rabbit-igg-hl-alexa-fluor-568-preadsorbed-ab175693.html visited on 2021-08-31.

[3] Urs Albrecht. Timing to perfection: The biology of central and peripheral circadian clocks. *Neuron*, 74(2):246–260, April 2012.

[4] Gheorghe Benga. On the definition, nomenclature and classification of water channel proteins (aquaporins and relatives). *Molecular aspects of medicine*, 33(5-6):514–517, 2012.

[5] Kristin Eckel-Mahan and Paolo Sassone-Corsi. Metabolism and the circadian clock converge. *Physiological Reviews*, 93(1):107–135, January 2013.

[6] Allen Institute for Brain Science. Allen brain atlas. Online, 2004. https://mouse.brain-map.org/static/atlas visited on 2021-08-31.

[7] DataBase Center for Life Science (DBCLS). Mouse during dissection. Online, 2013. https://commons.wikimedia.org/wiki/File:201308_mouse_during_dissection.svg visited on 2021-08-31.

[8] Patrick M. Fuller, Joshua J. Gooley, and Clifford B. Saper. Neurobiology of the sleep-wake cycle: Sleep architecture, circadian regulation, and regulatory feedback. *Journal of Biological Rhythms*, 21(6):482–493, December 2006. Publisher: SAGE Publications Inc.

[9] Gregory J. Gage, Daryl R. Kipke, and William Shain. Whole animal perfusion fixation for rodents. *Journal of visualized experiments : JoVE*, July 2012.

[10] Gwilz. Vector diagram of laboratory mouse (black and white). Online, 2013. https://commons.wikimedia.org/wiki/File:Vector_diagram_of_laboratory_mouse_(black_and_white).svg visited on 2021-08-31.

[11] Lauren M Hablitz, Virginia Plá, Michael Giannetto, Hanna S Vinitsky, Frederik Filip Stæger, Tanner Metcalfe, Rebecca Nguyen, Abdellatif Benrais, and Maiken Nedergaard. Circadian control of brain glymphatic and lymphatic fluid flow. *Nature communications*, 11(1):1–11, 2020.

[12] Joseph D Ho, Ronald Yeh, Andrew Sandstrom, Ilya Chorny, William EC Harries, Rebecca A Robbins, Larry JW Miercke, and Robert M Stroud. Crystal structure of human aquaporin 4 at 1.8 å and its mechanism of conductance. *Proceedings of the National Academy of Sciences*, 106(18):7437–7442, 2009.

[13] Kevin L Howe, Premanand Achuthan, James Allen, Jamie Allen, Jorge Alvarez-Jarreta, M Ridwan Amode, Irina M Armean, Andrey G Azov, Ruth Bennett, Jyothish Bhai, Konstantinos Billis, Sanjay Boddu, Mehrnaz Charkhchi, Carla Cummins, Luca Da Rin Fioretto, Claire Davidson, Kamalkumar Dodiya, Bilal El Houdaigui, Reham Fatima, Astrid Gall, Carlos Garcia Giron, Tiago Grego, Cristina Guijarro-Clarke, Leanne Haggerty, Anmol Hemrom, Thibaut Hourlier, Osagie G Izuogu, Thomas Juettemann,

Vinay Kaikala, Mike Kay, Ilias Lavidas, Tuan Le, Diana Lemos, Jose Gonzalez Martinez, José Carlos Marugán, Thomas Maurel, Aoife C McMahon, Shamika Mohanan, Benjamin Moore, Matthieu Muffato, Denye N Oheh, Dimitrios Paraschas, Anne Parker, Andrew Parton, Irina Prosovetskaia, Manoj P Sakthivel, Ahamed I Abdul Salam, Bianca M Schmitt, Helen Schuilenburg, Dan Sheppard, Emily Steed, Michal Szpak, Marek Szuba, Kieron Taylor, Anja Thormann, Glen Threadgold, Brandon Walts, Andrea Winterbottom, Marc Chakiachvili, Ameya Chaubal, Nishadi De Silva, Bethany Flint, Adam Frankish, Sarah E Hunt, Garth R IIsley, Nick Langridge, Jane E Loveland, Fergal J Martin, Jonathan M Mudge, Joanella Morales, Emily Perry, Magali Ruffier, John Tate, David Thybert, Stephen J Trevanion, Fiona Cunningham, Andrew D Yates, Daniel R Zerbino, and Paul Flicek. Ensembl 2021. *Nucleic Acids Research*, 49(D1):D884–D891, 11 2020.

[14] Jeffrey J Iliff, Minghuan Wang, Yonghong Liao, Benjamin A Plogg, Weiguo Peng, Georg A Gundersen, Helene Benveniste, G Edward Vates, Rashid Deane, Steven A Goldman, et al. A paravascular pathway facilitates csf flow through the brain parenchyma and the clearance of interstitial solutes, including amyloid β. *Science translational medicine*, 4(147):147ra111–147ra111, 2012.

[15] Jackson ImmunoResearch. Anti-goat igg h and l. Online, 2021. `https://www.jacksonimmuno.com/catalog/products/705-606-147` visited on 2021-08-31.

[16] Nadia Aalling Jessen, Anne Sofie Finmann Munk, Iben Lundgaard, and Maiken Nedergaard. The glymphatic system: a beginner's guide. *Neurochemical research*, 40(12):2583–2599, 2015.

[17] merckmillipore. Anti-aquaporin 4 antibody, ct. Online, 2021. `https://www.merckmillipore.com/CH/de/product/Anti-Aquaporin-4-Antibody-CT,MM_NF-AB3594-200UL?ReferrerURL=https%3A%2F%2Fwww.google.com%2F` visited on 2021-08-31.

[18] Yan Ouyang, Carol R Andersson, Takao Kondo, Susan S Golden, and Carl Hirschie Johnson. Resonating circadian clocks enhance fitness in cyanobacteria. *Proceedings of the National Academy of Sciences*, 95(15):8660–8664, 1998.

[19] Jonas Töle. Sagittal section of a mouse brain. Online, 2014. `https://commons.wikimedia.org/wiki/File:Mouse_brain_sagittal.svg` visited on 2021-08-31.

[20] Lulu Xie, Hongyi Kang, Qiwu Xu, Michael J Chen, Yonghong Liao, Meenakshisundaram Thiyagarajan, John O'Donnell, Daniel J Christensen, Charles Nicholson, Jeffrey J Iliff, et al. Sleep drives metabolite clearance from the adult brain. *science*, 342(6156):373–377, 2013.

[21] Y Xu, KL Toh, CR Jones, J-Y Shin, Y-H Fu, and LJ Ptáček. Modeling of a human circadian mutation yields insights into clock regulation by per2. *Cell*, 128(1):59–70, 2007.

[22] Binhai Zheng, David W Larkin, Urs Albrecht, Zhong Sheng Sun, Marijke Sage, Gregor Eichele, Cheng Chi Lee, and Allan Bradley. The mper2 gene encodes a functional component of the mammalian circadian clock. *Nature*, 400(6740):169–173, 1999.