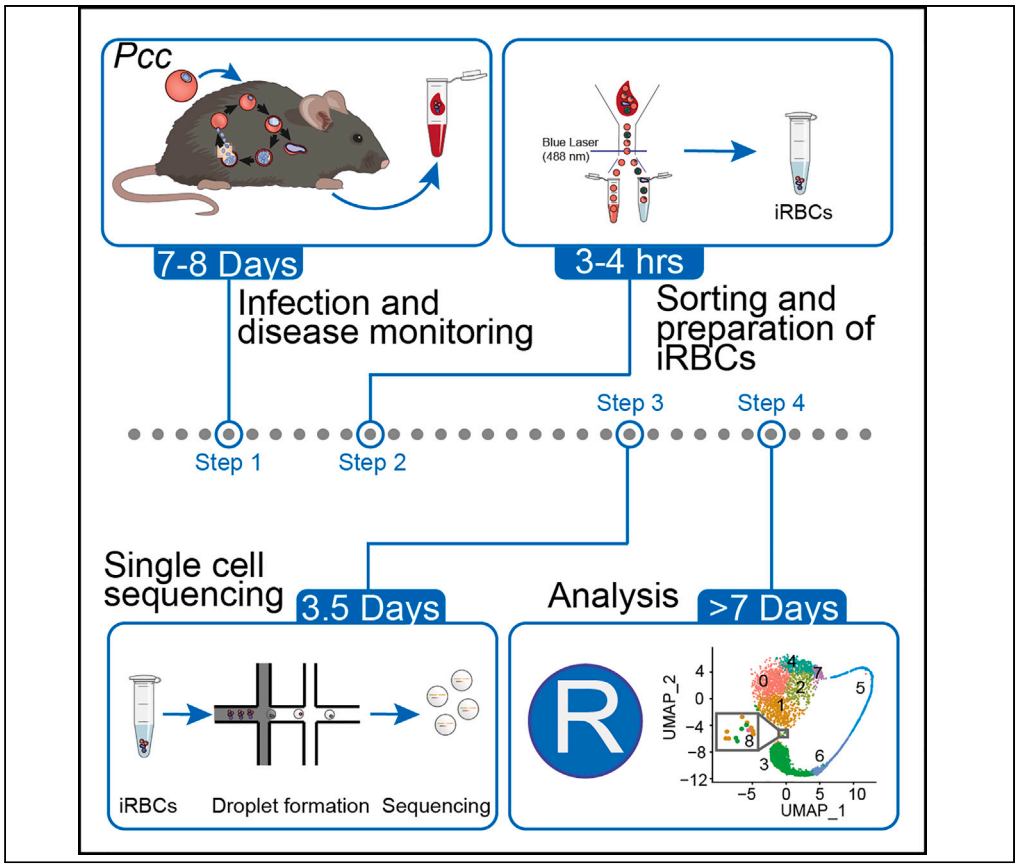
STAR Protocols

CellPress
OPEN ACCESS

Protocol

Single-cell RNA sequencing and analysis of rodent blood stage *Plasmodium*



Bulk RNA sequencing of *Plasmodium* spp., the causative parasite of malaria, fails to discriminate developmental-stage-specific gene regulation. Here, we provide a protocol that uses single-cell RNA sequencing of FACS-sorted *Plasmodium-chabaudi-chabaudi*-AS-infected red blood cells (iRBCs) to characterize developmental-stage-specific modulation of gene expression during malaria blood stage. We describe steps for infecting mice, monitoring disease progression, preparing iRBCs, and single-cell sequencing iRBCs. We then detail procedures for analyzing scRNA-seq data.

Publisher's note: Undertaking any experimental protocol requires adherence to local institutional guidelines for laboratory safety and ethics.

Elisa Jentho, António G.G. Sousa, Susana Ramos, ..., Ricardo B. Leite, Jingtao Lilue, Miguel P. Soares

ejentho@igc.gulbenkian.pt (E.J.)
aggode@utu.fi (A.G.G.S.)
mpsoares@igc.gulbenkian.pt (M.P.S.)

Highlights

ScRNA sequencing protocol for *Plasmodium chabaudi chabaudi* blood stages

Characterization of different *Plasmodium* blood stages by scRNA sequencing

Characterization of *Plasmodium* blood stage transitions by scRNA trajectories

Characterization of host effects on *Plasmodium* blood stages by scRNA sequencing

Jentho et al., STAR Protocols 4, 102491
September 15, 2023
https://doi.org/10.1016/j.xpro.2023.102491

STAR Protocols

CellPress
OPEN ACCESS

Protocol

Single-cell RNA sequencing and analysis of rodent blood stage *Plasmodium*

Elisa Jentho,[1,2,3,4,*] António G.G. Sousa,[1,3,4,*] Susana Ramos,[1] Temitope W. Ademolue,[1] João Sobral,[1] João Costa,[1] Denise Brito,[1] Marta Manteiro,[1] Ricardo B. Leite,[1] Jingtao Lilue,[1] and Miguel P. Soares[1,5,*]

[1]Instituto Gulbenkian de Ciência, Oeiras, Portugal

[2]Jena University Hospital, Department of Anesthesiology and Intensive Care Medicine, Friedrich-Schiller-University, Jena, Germany

[3]These authors contributed equally

[4]Technical contact

[5]Lead contact

*Correspondence: ejentho@igc.gulbenkian.pt (E.J.), aggode@utu.fi (A.G.G.S.), mpsoares@igc.gulbenkian.pt (M.P.S.)
https://doi.org/10.1016/j.xpro.2023.102491

SUMMARY

Bulk RNA sequencing of *Plasmodium* spp., the causative parasite of malaria, fails to discriminate developmental-stage-specific gene regulation. Here, we provide a protocol that uses single-cell RNA sequencing of FACS-sorted *Plasmodium-chabaudi-chabaudi*-AS-infected red blood cells (iRBCs) to characterize developmental-stage-specific modulation of gene expression during malaria blood stage. We describe steps for infecting mice, monitoring disease progression, preparing iRBCs, and single-cell sequencing iRBCs. We then detail procedures for analyzing scRNA-seq data.

For complete details on the use and execution of this protocol, please refer to Ramos et al.[1]

BEFORE YOU BEGIN

This protocol describes single cell sequencing of blood stages of the rodent-infective transgenic *Plasmodium chabaudi chabaudi AS* (*Pcc*) strain expressing the green fluorescent protein (GFP).[2] Specifically, inoculating mice with *Pcc* infected red blood cells (iRBC), bypasses the liver stage of the parasiteś lifecycle in its mammalian hosts. This approach is applicable to the analysis of blood stages of any transgenic *Plasmodium* strain expressing a fluorescent tag/protein.

Institutional permissions

All experimental procedures were approved by the Instituto Gulbenkian de Ciência Ethics Committee/ORBEA and the Portuguese National Entity (Direcção Geral de Alimentação e Veterinária). Experimental procedures were performed according to the Portuguese legislation on protection of animals and European (Directive 2010/63/EU) legislations. Animal experiments must be approved by the institutional ethics committee before the procedure and must be in accordance with national law on protection of animals.

Preparation of *Pcc* for infection

Timing: 7–10 days

Initial mouse infection

1. Thaw one stock vial of frozen mouse RBC infected with GFP-expressing *Pcc* (iRBC;1 × 10^6/100 μL).

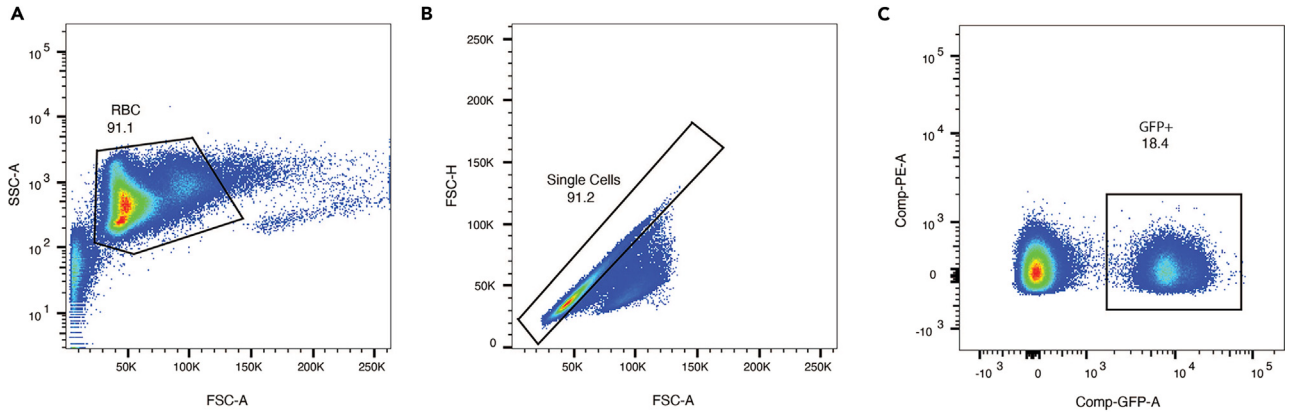


Figure 1. Gating strategy for parasitemia determination by flow cytometry

(A) RBC are identified in a FSC (cell size) vs. SSC (cellular granularity) plot. RBC constitute the main cell population in blood. The respective gate is designed to encompass most cells, excluding debris and high size or high granular events.

(B) Doubled exclusion identified by FSC (Hight) vs. FSC (Area) of RBCs.

(C) *Pcc*-infected RBC (iRBC) are identified inside the singlet population as defined in B, based on GFP expression. The percentage of iRBC (GFP^+) is recorded.

Numbers in (A–C) indicate percentages of gated events.

2. Use a 1 mL syringe with a 25G needle to inject iRBCs (200 μL) intra-peritoneally (i.p) into an adult (> 12 weeks old) C57BL/6J mouse.
3. Monitor the progression of the blood stage of infection, by assessing parasitemia (*i.e.*, % iRBC) daily from day 4 onwards, as described below.

   Note: Disease progression and severity are dependent on mouse genotype, age, and sex. *Pcc* AS infection is not lethal in C57BL/6J mice and the disease course is sex dependent, being more severe in males then females.

4. Parasitemia monitoring by flow cytometry:
   a. Prepare PBS (1×) freshly.
   b. Disinfect the tail with 70% ethanol prior blood collection.
   c. Collect a drop of blood (~2 μL) by puncturing the tip of the tail with a 27G needle. Discard the first drop of blood and collect the second drop into a polystyrene tube (1.5 mL) containing PBS (1×; 400 μL).
   d. Keep the samples at room temperature (22°C–24°C) if they will be assessed within 20 min after collection. If assessed later, store samples on ice or at 4°C. Samples can be assessed until 24 h after collection.
   e. Run the samples in a flow cytometry analyzer to monitor size by forward scatter (FSC), granularity by side scatter (SSC) and fluorescence, using a laser and bandpass detector suitable for the emission of the fluorescent tag/protein expressed by the transgenic parasite (*i.e.*, GFP in this manuscript).
   f. Analyze the sample in a size (FSC-A) *vs.* granularity (SSC-A) plot. RBC should correspond to the smaller and less granular cell compartment (Figure 1).
   g. Within the "RBC" population, define the region of the "GFP^+" population, typically identified in a bivariate graph plotting the signal from the blue laser (488 nm) detector equipped with a 530/30 bandpass optical filter (GFP expression) *vs.* the signal from the yellow-green laser (561 nm) detector equipped with a 586/15 bandpass optical filter (autofluorescence). This allows distinguishing and excluding auto-fluorescence from GFP^+ iRBC (Figure 1).
   h. Register the percentage of GFP^+ events inside the parent RBC gate to extrapolate parasitemia (% iRBC).
5. Parasitemia monitoring by microscopy:
   a. Collect a drop of blood onto a microscopy glass slide.

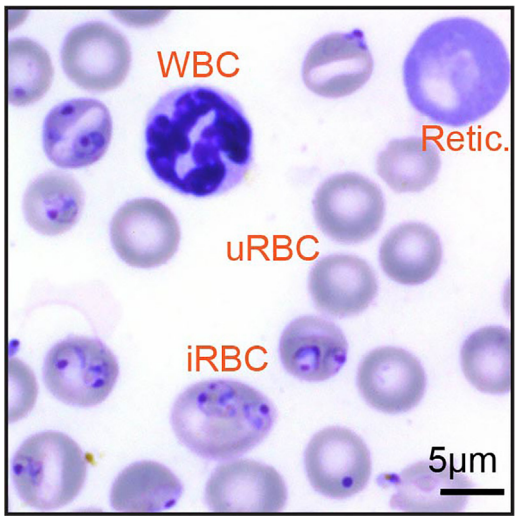


Figure 2. Giemsa-stained blood smear from *Pcc*-infected mouse

Blood was collected from the tail of a *Pcc*-infected mouse, fixed, and stained with Giemsa (10%). WBC: white blood cell, e.g., a neutrophil, distinguishable by size and DNA content and nuclei shape. Retic.: Reticulocyte, immature RBC, bigger than mature RBC with remnants of nucleic acids scattered throughout the cytoplasm. uRBC: uninfected mature RBC. iRBC: *Plasmodium*-infected RBC. Scale bar: 5 μm.

b. Use another glass slide, placed in a 45° angle, and smear the blood through the slide.
c. After the blood smear dried, fix it with methanol (100%; by soaking in methanol for a couple of seconds)
d. Take the glass slide out of the methanol and let it dry on air.
e. Stain the slide with freshly prepared Giemsa solution (10% in water) for 10 min at 22°C. This will stain nucleic acids of the parasite since mature RBC do not contain any nucleus.
f. Wash the slide in running water.
g. Let the slide dry at 22°C (min. 10 min) and analyze the smear under the microscope. Put a drop of mineral oil directly on top of the smear and analyze it using a 100× magnification objective. Parasites should be readily identifiable inside the RBC. White blood cells should also be easily identifiable by its size (usually bigger than RBC) and DNA content, occupying most of the cell (Figure 2).

Note: After drying the slide the sample is stable and can be analyzed at a later time point. In principle the slides can be stored (22°C–24°C) and analyzed months after the staining.

h. Count the number of RBC containing parasites (iRBC) and total RBC *per* field in 4 fields. Calculate parasitemia as the percentage of iRBC:

$$\%iRBC = \frac{number\ of\ iRBC}{number\ of\ totalRBC} x100$$

Experimental mice: Infection

Timing: 0.5 days

6. Once parasitemia reaches 10%–20% (usually around day 8–10 after iRBC inoculation) proceed to infect an experimental C57BL/6J mouse.
7. Count reference latex beads at the microscope in a Neubauer chamber.
   a. Assemble the Neubauer chamber with the cover slide.
   b. Add bead suspension (10 μL) one edge of the glass cover, so that through capillary forces the liquid fills the chamber.
   c. Count beads in the 4 large corner squares and calculate the concentration as follows:

$$\text{Beads} \Big/ \text{mL} = \frac{Total\ number\ of\ counted\ beads\ x\ 10.000}{Number\ of\ squares\ counted}$$

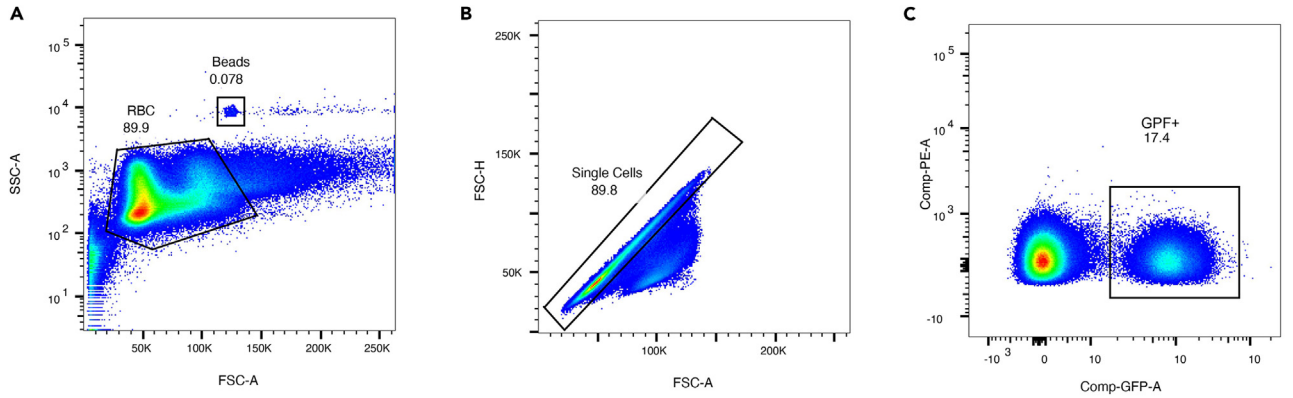


Figure 3. Gating strategy for parasitemia determination and cell counting by flow cytometry

(A) RBC are identified in a FSC (cell size) vs. SSC (cellular granularity) plot. RBC constitute the main blood cell population. The respective gate is designed to encompass most cells, excluding debris and high size or high granularity events. Latex beads are easily identifiable, having a higher granularity, compared to blood cells (i.e., higher SSC).

(B) Doubled exclusion identified by FSC (Hight) vs. FSC (Area) of RBCs.

(C) *Pcc*-infected RBC (iRBC) are identified in the RBC population defined in B, based on GFP expression. The percentage of iRBC (GFP^+) is recorded, as well as the event counts of GFP^+ and beads.

Numbers in (A–C) indicate percentages of gated events.

8. Prepare a polystyrene tube (1.5 mL) containing PBS (1×; 400 μL) and reference latex beads (10 μL).
9. Collect blood (2 μL, as described in step 4) from the infected mouse into the tube containing PBS and beads.
10. Run the sample in a flow cytometry analyzer.
11. Use the same gating strategy as described above (Figure 1).
12. Define an additional gate identifying the beads according to FSC *vs.* SSC plot (Figure 3).
13. Acquire a total of 400 events in the beads gate.
14. Save and export the number of total GFP^+ iRBC and beads, as well as the percentage of iRBC.
15. Calculate the number of iRBC *per* μL of blood using the following equation:

$$RBC\ per\ \mu L\ =\ \frac{GFP\ events\ x\ Beads\ concentration\ xDilution\ factor}{\text{"}Beads\text{"}events}$$

Note: The Dilution factor refers to the ratio of beads to blood volume, i.e., 10 μL–2 μL = 5. Of note, using a flow cytometer that measures sample volume, this step is bypassed, and the only requirement is to correct for the dilution of blood in 1× PBS, *i.e.*, 2 μL in 400 μL.

16. Calculate the volume of *Pcc*-infected mouse blood required to prepare a solution containing 1×10^7 *Pcc*-iRBC/mL in PBS (1×):

$$Volume\ of\ Pcc\ infected\ blood\ =\ \frac{1x10^7 Final\ iRBC\ per\ \mu L\ x\ Final\ volume}{Calculated\ iRBC\ per\ \mu L}$$

Note: The final volume will depend on the number of experimental mice, considering that an injection volume of 200 μL *per* mouse should be used.

17. Harvest the calculated amount of blood volume from the superficial facial temporal vein of the infected mouse by puncturing with a 25G needle. All following steps are performed at RT.
18. Collect blood into a polystyrene tube (1.5 mL) containing heparin (25.000 UI/5 mL; 2 μL).
19. Alternatively collect blood by cardiac puncture after euthanasia using CO_2.
    a. Coat a 1 mL syringe with a 25G needle with heparin (25.000 UI/5 mL), by filling the syringe with heparin and ejecting it back into the heparin flask. The residual heparin in the syringe and needle is enough to avoid blood clotting.

b. Perform cardiac puncture according to previously described methods.[3]
c. Collect blood into a polystyrene tube (1.5 mL) containing heparin (25.000 UI/5 mL; 10 µL).

20. Dilute blood in the calculated final volume of PBS (1×).
21. Inject 200 µL (i.p.) into each "experimental" mouse.

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Chemicals, peptides, and recombinant proteins		
Giemsa's stain improved r66 soln. Gurr	VWR	Cat# 350864X CAS: 51811-82-6
10× PBS	GRiSP Research Solutions	Cat# GS11.0110
Cell culture grade water	GRiSP Research Solutions	Cat# GTC99.1000
Methanol, p.a. Normapur	VWR	Cat# VWRC20847.307
Heparina Leo 25.000 µL/5 mL	Leo Pharma A/S	Cat# 014425-03
EDTA Buffer, pH 8, 0.5 M	Thermo Fisher Scientific	Cat# 15575020
FBS, Qualified 500 mL	Thermo Fisher Scientific	Cat# 10270106
Critical commercial assays		
Chromium Next GEM Single Cell 3' GEM, Library & Gel Bead Kit v3.1, 4 rxns	10× Genomics	Cat#1000128
Chromium Next GEM Chip G Single Cell Kit, 16 rxns	10× Genomics	Cat#1000127
Chromium i7 Multiplex Kit, 96 rxns	10× Genomics	Cat#120262
NextSeq 500/550 High Output Kit v2.5 (150 Cycles)	Illumina	Cat# 20024907
Qiagen Buffer EB	Qiagen	Cat#50919086
HS NGS Fragment kit	Agilent Technologies	Cat#DNF-474
D1000 ScreenTape	Agilent Technologies	Cat#5067-5582
D1000 Reagents	Agilent Technologies	Cat#5067-5583
D1000 Ladder	Agilent Technologies	Cat#5067-5586
Deposited data		
Single Cell RNA-seq analysis of *Plasmodium* parasites		Array Express: E-MTAB-10939
Experimental models: Organisms/strains		
Mouse: *B6*.C57BL/6/J, 12–14 weeks, male or female	The Jackson Laboratory	N/A
Plasmodium chabaudi chabaudi strains: *Pcc*AS-GFP$_{ML}$ Passage higher than 17.	Marr et al.[2] Kind gift from Dr. J. Thompson	N/A
Software and algorithms		
FlowJo	FlowJo	https://www.flowjo.com/solutions/flowjo; V10
BD FACSDiva™ FACSDiva software v 8.0.3	BD Biosciences	https://www.bdbiosciences.com/en-tw/products/instruments/software-informatics/instrument-software/facs-diva/facsdiva-software-v-8-0-3-upgrade-win-7-32-bit-os.659528; V8.0.3
STAR	Dobin et al.[4]	https://github.com/alexdobin/STAR ; v.2.7.3a
R programming language and environment	R Core Team, 2020 http://subread.sourceforge.net/	https://www.r-project.org/ ; v.3.6.3
Rstudio Server	© 2009–2020 Rstudio, PBC	https://www.rstudio.com/products/rstudio/ ; v.1.3.1093
ggplot2	Wickham[5]	https://cran.r-project.org/web/packages/ggplot2/index.html ; v.3.3.2

(*Continued on next page*)

Continued

REAGENT or RESOURCE	SOURCE	IDENTIFIER
ComplexHeatmap	Gu et al.[6]	https://bioconductor.org/packages/release/bioc/html/ComplexHeatmap.html ; v.2.2.0
gprofiler2	Kolberg et al.[7]	https://cran.r-project.org/web/packages/gprofiler2/index.html ; v.0.2.0
Cell Ranger	10× Genomics	https://support.10xgenomics.com/single-cell-gene-expression/software/pipelines/latest/what-is-cell-ranger ; v4.0.0
Seurat	Hao et al.[8]	https://cran.r-project.org/web/packages/Seurat/index.html ; v.4.0.0
velocyto	La Manno et al.[9]	http://velocyto.org/v.0.17.17
GNU parallel	https://doi.org/10.5281/zenodo.1146014	https://www.gnu.org/software/parallel/v.20161222
scvelo	Bergen et al.[10]	https://scvelo.readthedocs.io/; v.0.2.2
anndata		https://github.com/theislab/anndata/v.0.7.5
pandas	Team[11]	https://pandas.pydata.org/ v.1.1.3 and v1.2.4
numpy	Harris et al.[12]	https://numpy.org/v.1.19.2, v1.20.3
matplotlib	Hunter[13]	https://matplotlib.org/v.3.3.2
python	Python Software Foundation	https://www.python.org/v.3.8.3
scipy	Virtanen et al.[14]	https://scipy.org/citing-scipy/ v1.6.3
Other		
Accu-chek performa blood glucose meter and 50ct strip	Roche	Cat# 6454011052
BD FACSCalibur™	BD Biosciences	
BD LSRFortessa™ BD LSRFortessa X-20 Cell Analyzer	BD Biosciences	Cat# 65767502
BD FACSAria™ Iiu Cell Sorter	BD Biosciences	
Rodent thermometer	BioSET	Cat# BIO-TK8851
Chromium™ Single Cell Controller	10× Genomics	Cat# 120263
C1000 Touch Thermal Cycler with 96-Deep Well Reaction Module	Bio-Rad	Cat# 1851197
Fragment Analyzer Automated CE System	Agilent Technologies	Cat#FSv2-CE2F
4200 TapeStation System	Agilent Technologies	Cat# G2991BA
NextSeq 500 Sequencing System	Illumina	Cat#GEILSY-415-1001
Sterilized needles 16–5 /10–25Gx5/8	Terumo	Cat# TERUAN-2516R1
Syringes (without needle 1 mL)	VWR	Cat# 613-2041
PCR tubes (0.2 mL) 8-tube strips	Eppendorf	Cat#732-0098
Tube DNA Lobind (1,5ML)	Eppendorf	Cat# 10051232
Polystyrene tube (1.5 mL)	Sarstedt	Cat# 72.690.550
Slide Beveled Edges Frosted 76 × 26MM	VWR	Cat# VWRI631-1553
Light microscope		
Standard, L10, 10 µM, Latex Particle	Beckman Coulter	Cat# 0826602796
Tube with conical bases, PP, 120 × 17 mm, 15 mL, red screw cap	Sarstedt	Cat# 62.554.502/250
BRAND® counting chamber BLAUBRAND® Neubauer improved	Merck	Cat# BR717805-1EA
Bright-Line™ Hemacytometer replacement cover slip	Sigma-Aldrich	Cat# Z375357-1EA
Laboratory bottles, narrow neck, with screw cap, DURAN® - 100 mL	VWR	Cat# 215-1514
Cytiva Whatman™ Qualitative Filter Paper: Grade 1 Circles	Fisher Scientific	Cat# 10424081

MATERIALS AND EQUIPMENT

Note: Buffers are prepared freshly.

Giemsa Solution		
Reagent	Final concentration	Amount
Giemsa's stain solution	10%	10 mL
Buffered Water (pH 7.2)	N/A	90 mL

- Dissolve 1 buffer tablet in 1 L of distilled water.
- Mix the Giemsa's stain (v/v) with the buffered water.
- Filter the solution using Whatman filter papers, placed on a small funnel.

1× PBS		
Reagent	Final concentration	Amount
10× PBS	1×	5 mL
Cell culture grade water	N/A	45 mL

1× PBS supplemented with 1% FBS		
Reagent	Final concentration	Amount
10× PBS	1×	5 mL
Cell culture grade water	N/A	45 mL
FBS	1%	500 µL

STEP-BY-STEP METHOD DETAILS

Plasmodium chabaudi chabaudi infection and disease monitoring

Timing: 7–8 days

In this section we describe *Pcc* infection and disease monitoring in C57BL/6J mice (12–14 weeks old).

The virulence of *Pcc* infection varies according to the number of times the parasites are passaged in mice, i.e., when the parasite life cycle stages naturally occurring in the mosquito and mouse liver are bypassed and mice are infected by sequential *Pcc*-infected blood transfusions.[15] The first blood stage infection derived from the mosquito (i.e., passage 0, P0) is established after a liver stage of infection, and is characterized by the development of low parasitemia, rapid parasite clearance and low virulence, i.e., slight deviation in health parameters. As the number of blood passages increases, parasitemia and virulence increase accordingly. The values presented below for health parameters and parasitemia refer to 7 and ~30 passages (P30) after the initial mosquito bite and liver infection (Figure 4).

Note: Depending on the parasite and mouse strain used, time and severity of infection might differ. It is recommended to perform a disease progression analysis prior to the single-cell RNA sequencing experiment to identify the time point of interest. Protocol is described for adult male C57BL/6J mice maintained under specific pathogen free conditions (SPF), fed ad libitum with standard chow and maintained at 22°C room temperature with a light cycle of 12/12 h.

1. Disease monitoring: Monitor mouse health parameters before the infection (Day 0) to define steady state and to calculate deviations from steady state during the course of infection.

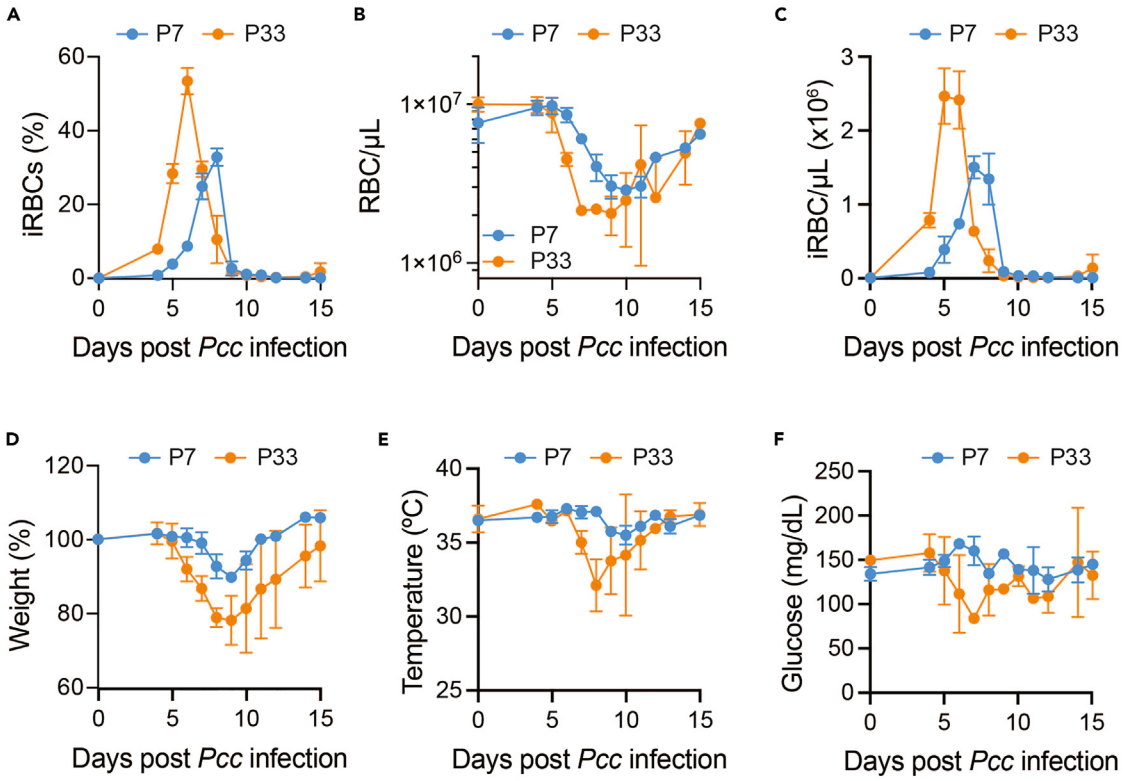


Figure 4. Disease progression in *Pcc*-infected C57BL/6J mice

Mice were infected with different passages of *Pcc* AS parasites (P7, P33). Disease and parasite development were monitored daily for 15 days. (A–F) (A) Parasitemia, i.e., percentage of infected RBC (iRBC); (B) number of RBC *per* µL of blood (RBC/µL); (C) pathogen load, i.e., number of iRBC RBC *per* µL of blood (iRBC/µL); (D) weight loss, i.e., percentage of initial weight; (E) core body temperature; (F) glycemia. N = 2 mice per group. Data as mean ± SD in all plots.

a. Core body temperature is measured using a rectal probe. Steady state values should range between 36°C and 38°C.
b. Body weight is measured using a scale. Steady state values should range between 24–28 g for male C57BL/6J mice, at 12–14 weeks of age.
c. Glycemia is measured in a drop of blood, collected as described for parasitemia determination using a Glucometer. Steady state values should range between 120–180 mg/dL.

Note: Steady state core body temperature and glycemia measurements are highly dependent on the mouse's stress status and vary accordingly. It is advisable to handle mice a few days, before starting the experiment, for acclimatization to manipulation, to prevent misleading steady-state measurements. Core body temperature and glycemia also vary in a circadian manner[16,17] and as such it is advisable to monitor mice at same time of the day.

2. *Pcc* infection: Infect mice as described above in the experimental mice infection section. The day of infection is considered as day 0 (D0).
3. Disease progression monitoring: Monitor mice daily, from day 3 onwards, for the parameters described above:

Note: Core body temperature of *Pcc*-infected male C57BL/6J mice can decrease up to 29°C–32°C, at day 7 post infection (*i.e.*, peak of infection), corresponding to a loss of ~6°C–9°C (10%–20%), relative to steady state. This is readily reversed in the days following the peak of infection, typically by day 9 all mice should have recovered normal core body temperature.

Body weight of *Pcc*-infected male C57BL/6J mice can decrease up to 17–20 g at the peak of infection, corresponding to a loss of 5–8 g (20%–30%) of the initial body weight. Body weight recovery can be observed upon parasite clearance, typically at days 9–10 post-infection. Glycemia of *Pcc*-infected wild type C57BL/6J mice usually decreases up to 60–100 mg/dL at the peak of infection, corresponding to a loss of 60–120 mg/dL (50%–70%), relative to steady state. This represents a mild hypoglycemia, which is rapidly reversed after the peak of infection. Parasitemia: as described above (including flow cytometry and microscopical analysis). Typically, *Pcc* infection develops slowly in the first days, gaining momentum around D4-D7, where the percentage of iRBC can reach up to ~60%. Day 7 of infection is considered the peak of infection, after which parasites start to be cleared from circulation and parasitemia starts to decrease.

⚠ CRITICAL: The values presented here refer to male C57BL/6J mice infected with *Pcc* AS. These values are expected to vary according to the genetic background of the infected mice[18] as well as in genetically-modified C57BL/6J mice. As an example of the latter, genetic loss-of-function of glucose 6 phosphatase c (*G6pc1*) specifically in hepatocytes is associated with the development of severe hypoglycemia (<40 mg/dL) as well as defective adaptive thermoregulation (24°C–28°C), leading to death.[1]

⚠ CRITICAL: Mice should be monitored at the same time of the day, as some rodent *Plasmodium* strains, including *Pcc*, have a highly synchronous 24 h life cycle. This imparts that the parasite develops through the different life-cycle stages under circadian regulation. This in turn implies that disease severity also changes according to the hour of the day, which should be taken into consideration in the experimental design.

Sorting and preparation of iRBCs for single-cell RNA sequencing

Timing: 3–4 h

Here we describe how to prepare the iRBCs for single-cell RNA sequencing. This should be performed at the time-point of interest (e.g., when host or parasite present a specific phenotype) as determined by disease progression, as well as by visual observation of *Plasmodium* parasites in Giemsa-stained blood smears, monitored as described above.

⚠ CRITICAL: RBCs are very sensitive to shear stress. Accordingly, pipetting should be avoided whenever possible. If necessary, pipetting should be performed very gently to avoid damaging the RBC. Cell viability and transcriptional profiles are affected by the time of storage on ice. While parasitemia can be assessed up to 24 h after blood collection, the transcriptional profile might change significantly over time. Accordingly, it is advisable to start the single cell analysis as soon as possible after the sample preparation ideally, within 2 h after cell sorting. Low-binding plasticware is recommended for all the following steps regarding 10× genomics libraries.

4. At the peak of infection (i.e., 6–8 days after inoculation with iRBC), blood is collected by cardiac puncture, as follows:
   a. To euthanize mice by CO_2 exposure, place the animal in a container, connected to a CO_2 source allowing for air plus CO_2 gas exchange (e.g., containing a vent that can be opened and closed).
   b. Start with a slow flow rate (3 L/min) of CO_2 and an open vent to anaesthetize the mouse.
   c. Close the vent once the mouse is anaesthetized and increase the flow rate (7 L/min) until no breathing of the mouse is observed.

Note: Euthanasia can be induced in different ways, for example by Ketamine/Xylazine. It should, however, be consistent between experiments to avoid variability.

d. Confirm death by squeezing the hind paw. If there is no reaction proceed immediately with the cardiac puncture.
   i. Prior to the cardiac puncture, coat a 1 mL syringe with 0.5 M EDTA (RT), by filling the syringe with EDTA (without the needle) and then placing the needle (23G) on the syringe to eject the EDTA back into the original flask. The residual EDTA in the syringe and needle is enough to avoid blood clotting.
   ii. Perform cardiac puncture according to previously described methods.[3]
   iii. Transfer blood into a polystyrene tube (1.5 mL) and place it on ice. Sorting should be started as soon as possible but not later than 1h after blood collection.

5. Prior to sorting, confirm parasitemia, as described in the preparation of *Pcc* infection.
6. Dilute blood 1/75 vol/vol in PBS (1×) supplemented with FBS (1%). Prepare the dilution in a FACS tube to avoid multiple pipetting steps.

Note: Cut the end of the pipette tip when pipetting blood to avoid RBC lysis. When using a 70 μm nozzle for cell sorting, RBC concentration must be adjusted to ensure the threshold rate is not higher than 22,000 events *per* second.

7. Sort iRBCs according to FSC-A, SSC-A and GFP signals, as detailed above for parasitemia determination.
8. Collect sorted cells into polystyrene tubes (15 mL) containing 1 mL PBS (1×) supplemented with FBS (1%) to minimize cell lysis.
   a. Adjust the sensitivity of the FSC and SSC detectors to clearly visualize RBCs in the scatter plot. Define a region that includes RBCs but excludes white blood cells and debris (Figure 1).
   b. Display the RBCs in a new plot with FSC-A *vs.* FSC-H and/or FSC-A *vs.* FSC-W to exclude doublets, which have a higher, disproportional area, when compared with the second parameter.
   c. On a third bivariate plot, visualize the GFP fluorescence of the "singlet" events previously gated out of the blue laser (using a BP530/30 optical filter), against an empty detector on the yellow region out of the violet, blue or yellow green lasers (using a BP586/15 optical filter).
   d. Identify the GFP^+ events and define the region of the iRBC, excluding cell autofluorescence, which can be visualized in the empty detector.
   e. Use the defined gate to sort iRBCs using a Purity sorting mode.

⚠ CRITICAL: Do not set the sorting gate in a histogram plotting GFP signal, as this may lead to contamination of the sorted sample with GFP^- cells.

   f. To obtain an adequate number (> 1.5×10^6) of iRBC, sort samples for at least 30 min.
9. Centrifuge samples at 300 × *g*, 4°C, 5 min. Pellet should resemble the one illustrated in Figure 5.
10. Discard the supernatant into a waste bin by inverting the tube. Avoid aspirating the supernatant by a vacuum pump as this can increase RBC loss.
11. Wash the sample by adding cold PBS (1×, 2 mL).
12. Re-suspend the RBC pellet by inverting and flicking the tube. In case pipetting is needed, use a 1000 μL pipette and cut the pipette tip so that the opening is wider.
13. Repeat step 7–8 once.
14. Resuspend the cells in PBS (1×; 1 mL)
15. Count RBC using a Neubauer counting chamber (Figure 6).
   a. Prepare the Neubauer chamber by placing a glass cover on the chamber.
   b. Add the cell suspension (10 μL) on one edge of the glass cover, so that the liquid fills the chamber through capillarity. If air bubbles occur, disassemble the Neubauer chamber, clean it with ethanol (75%) and repeat the above-described steps.
   c. Place the chamber under the microscope and count the cells of the 4 large corner squares.
   d. The total number of RBC is calculated as follows:

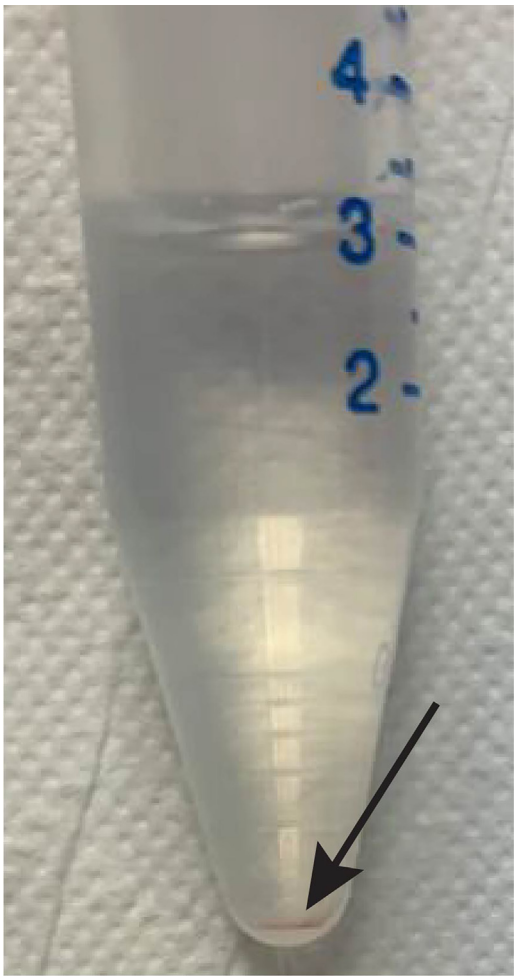


Figure 5. Illustration of the sample before counting the cells

$$\text{Cells} \Big/ \text{mL} = \frac{Total number of counted cells x 10.000}{Number of squares counted}$$

16. Repeat step 7–8 and re-suspend the sample in cold PBS (1×) to reach a final concentration of 1200 cells/μL.

10× genomics single-cell sequencing of iRBC

Timing: 3.5 days

The workflow, from cell encapsulation to library construction, is executed according to the recommended user guide (CG000204 Rev D – Single Index; https://www.10xgenomics.com/support/single-cell-gene-expression/documentation/steps/library-prep/chromium-single-cell-3-reagent-kits-user-guide-v-3-1-chemistry) retrieved on January 10th 2023.

17. Add the cell suspension (13.8 μL) to the prepared master mix (according to the user guide), keep on ice, to target a recovery of >10,000 cells (13.8 μL x 1200 cells/μL).

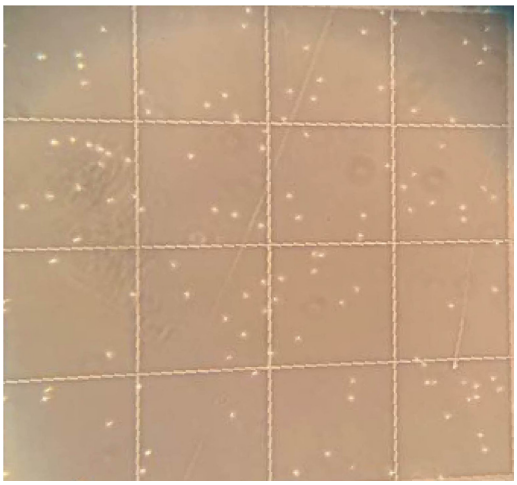


Figure 6. Neubauer chamber with iRBCs under the microscope
Magnification 20×.

18. Load the master mix containing the cells, the unique molecular identifier (UMI) and partition oil onto Chromium Next GEM Chip G (Chromium™ Single Cell Controller running Firmware 5.0 10× Genomics, PN-120263).
19. Perform reverse transcription of RNA into cDNA.
20. Analyze the cDNA profile on a Fragment Analyzer System (Agilent Technologies) according to the High Sensitivity NGS Fragment kit instructions (Agilent Technologies).

Note: This step is a crucial quality control for all subsequent procedures. The number of PCR cycles for cDNA amplification are adjusted according to the input cDNA mass on 10 µL of the total reaction, avoiding overamplification and possible artifacts. Library quality profile is also verified on the same instrument for both size determination and library quantification.

21. Sequence with NextSeq 500 (Illumina) using the NextSeq 500/550 High Output Kit v2.5 (150 Cycles) using a final diluted pool 1.8 pM with the following sequencing run settings: Read1: 28 cycles, i7 index: 8 cycles, i5 index: 0 cycles, Read2: 130 cycles, aiming a read depth of ~25,000 reads/cell.
22. Increase Read2 from the recommended 91 cycles to 130 due to cycle availability of the kit and to increase the mapping of the read to the reference transcriptome.

Analysis of scRNA-Seq data

Timing: > 2 weeks

Here we describe how to analyze the single-cell RNA sequencing data obtained with the protocol described above. In this section, the protocol starts by importing the filtered feature-barcode matrices, bad-quality cells and non-expressed genes are excluded, and the processed data is clustered into transcriptionally similar populations and projected onto the low-dimensional space. The cell populations identified are confirmed by inspecting conserved markers identified through differential gene expression (DGE).

Note: Sequence alignment: Feature Count Matrix is created using Cellranger V7.0.0 (10× Genomics) with STAR aligner V2.7.9a on an Ubuntu virtual machine. The reference genome of *Pcc* and annotation (Ensembl ftp://ftp.ensemblgenomes.org/pub/protists/release-50/protists/) is used for alignment through the "mkref" command from the Cellranger analysis pipeline. Raw

reads from each sample and the reference genome prepared on the previous step were processed with the "cellranger count" pipeline resulting in the Counts Matrix for each sample. All the steps are performed following standard best practices from 10× Genomics (https://support.10xgenomics.com/single-cell-gene-expression/software/pipelines/latest/what-is-cell-ranger). QC, filtering and exploratory clustering, dimensional reduction and DEG: All following analyses are performed in a virtual machine with 10 cores and 64 GB of RAM running Ubuntu 18. All the R code are run in RStudio Server (v. 2022.07.1 Build 554) using R version 3.6.3[11] as Rmarkdown notebooks (v.2.5[19,20]). Code described herein to perform data analyses was deposited in the following github repository: https://github.com/inflammationlab/scRNAseq-malaria-STAR-protocol in the hope of making analyses reproducible and reusable as well as to simplify the descriptions made herein which will be circumscribed to the most important steps for readability.

23. Single-cell analyses in R are performed using the Seurat package (v.4.0.0[8]). The Seurat package is forked and the 'DESCRIPTION' file modified (see: https://github.com/antonioggsousa/seurat/commit/3a9e57b499e9e417f09da3ebb0aeaa4cedfb9ef8) to install Seurat v.4.0.0 in R v.3.6.3. The installation is performed after launching the R console as follows:

```
> devtools::install_github(''antonioggsousa/seurat'')
```

24. Each single-cell sample, hereby labeled as gt1 and gt2, are analyzed independently and the differences in parameters are highlighted be.
25. Import the three files - 'barcodes.tsv.gz', 'features.tsv.gz', 'matrix.mtx.gz' - under the output folder 'filtered_feature_bc_matrix' comprising the filtered feature-barcode matrices MEX obtained in the previous step as a sparse matrix which was in turn converted into a S4 Seurat object. This step is done using two Seurat (v.4.0.0[8];) functions as described below:

```
>set.seed(seed = 1024) # to keep reproducibility

>library("Seurat", quietly = TRUE) # import package

# import 10x Gt1 sample

>gt1_sample_dir <- "../data/ftp01.igc.gulbenkian.pt/gt1_count_full/outs/filtered_feature_bc_matrix"

>gt1 <- Read10X(data.dir = gt1_sample_dir)

>gt1_seu <- CreateSeuratObject(counts = gt1, project = "Gt1", min.cells = 3) # create Seurat object

# import 10x Gt2 sample

>gt2_sample_dir <- "../data/ftp01.igc.gulbenkian.pt/Gt2_count_full/outs/filtered_feature_bc_matrix"

>gt2 <- Read10X(data.dir = gt2_sample_dir)

>gt2_seu <- CreateSeuratObject(counts = gt2, project = "Gt2", min.cells = 3) # create Seurat object
```

26. Remove genes expressed in less than three cells ('CreateSeuratObject(..., min.cells = 3')).
27. Inspected different single-cell data features in terms of the number of total UMIs (Unique Molecular Identifiers) or different number of genes expressed *per* cell.
28. Remove Cells expressing less than 750 different genes or 1,500 UMIs.

```
>params <- list("max_nFeature" = 750, "max_nCount" = 1500) # parameters to apply the filtering

>gt1_seu <- subset(gt1_seu, subset = nFeature_RNA < params$max_nFeature & nCount_RNA
< params$max_nCount) # apply the filtering

>gt2_seu <- subset(gt2_seu, subset = nFeature_RNA < params$max_nFeature & nCount_RNA
< params$max_nCount)
```

Note: By looking into its distribution, e.g., using the function 'VlnPlot(gt1_seu, features = c("nFeature_RNA", "nCount_RNA"))', and their linear relationship, e.g., plotting 'FeatureScatter(gt1_seu, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")' other cell features that might indicate cell integrity are often used as well as filtering criteria such as the percentage of expression of mitochondrial and/or ribosomal genes.[21] These features were not considered in this protocol as they are difficult to disentangle from the distinct life-cycle stages where the variation in gene expression of mitochondrial/ribosomal genes is likely to happen. These thresholds fit above the upper whisker (outliers) when the distribution of these features is plotted, likely representing doublets or multiples. Cells expressing low number of genes or UMIs might represent non-intact cells, dying cells or cells with broken membranes, and they should be excluded as well at this stage (e.g., 'subset(gt1_seu, subset = nFeature_RNA > 100 & nCount_RNA > 150)'). As the number of genes detected and UMIs obtained for the different Plasmodium life-cycle stages diverges,[22] and smaller cells have lower mRNA content[23] thresholds may not be applied or applied with caution as they may exclude resting, less active stages. The thresholds applied always need to be adapted to each study by looking into the distribution of the different cell features.

29. Filtered gene expression count tables are normalized with the function: 'NormalizeData()' using the 'normalization.method = "LogNormalize"' and 'scale.factor = 10000'.

```
>gt1_seu <- NormalizeData(gt1_seu, normalization.method = "LogNormalize", scale.factor =
10000)

>gt2_seu <- NormalizeData(gt2_seu, normalization.method = "LogNormalize", scale.factor =
10000)
```

30. Select the top 750 highly variable features across the datasets with the function 'FindVariableFeatures()' using the following parameters: 'selection.method = "vst"' and 'nfeatures = 750'.

```
> gt1_seu <- FindVariableFeatures(gt1_seu, selection.method = "vst", nfeatures = 750) # find
variable features

>gene_names <- rownames(gt1_seu) # retrieve gene names 4554

>gt1_seu <- ScaleData(object = gt1_seu, features = gene_names) # run scaling

>gt2_seu <- FindVariableFeatures(gt2_seu, selection.method = "vst", nfeatures = 750) # find
variable features

>gene_names <- rownames(gt2_seu) # retrieve gene names 4554

>gt2_seu <- ScaleData(object = gt2_seu, features = gene_names) # run
```

Note: Usually, 2k features are chosen for data sets with more than 10k genes, which represents less than 20% of the genes. Therefore, 750 variable features are chosen to represent ~16% of

all variable features for *Pcc* samples. Then all the features are scaled with the 'ScaleData()' function.

31. A Principal Component Analysis (PCA) is performed using the scaled top highly variable genes with the function 'RunPCA()'. This PCA is in turn used to cluster and build the non-linear Uniform Manifold Approximation and Projection (UMAP) later.

```
> gt1_seu <- RunPCA(object = gt1_seu, features = VariableFeatures(object = gt1_seu)) # run PCA
>gt2_seu <- RunPCA(object = gt2_seu, features = VariableFeatures(object = gt2_seu)) # run PCA
```

32. The importance of the different principal component regarding the contribution of variance is inspected through an elbow plot and based on the result the top 10 PCs were selected for clustering and UMAP projection (Figure 7).

```
>library(''ggplot2'')
>elbow_plot <- ElbowPlot(gt1_seu, ndims = 50) +
theme_minimal() +
geom_vline(xintercept = 10, linetype = "dashed") +
theme(axis.text = element_text(size = 14, color = "black"),
axis.title = element_text(size = 14, color = "black"),
  text = element_text(size = 14, color = "black")) +
annotate("rect", xmin = 0, xmax = 10, ymin = 0, ymax = 10,
alpha = .25, fill = "#E64B35FF") +
annotate(geom = "text", x = 5, y = 7.5, label = paste0("10
PCs"), size = 4.5, color = "blue")
>print(elbow_plot) # print
>elbow_plot <- ElbowPlot(gt2_seu, ndims = 50) +
theme_minimal() +
geom_vline(xintercept = 10, linetype = "dashed") +
theme(axis.text = element_text(size = 14, color = "black"),
axis.title = element_text(size = 14, color = "black"),
text = element_text(size = 14, color = "black")) +
annotate("rect", xmin = 0, xmax = 10, ymin = 0, ymax = 10,
alpha = .25, fill = "#E64B35FF") +
annotate(geom = "text", x = 5, y = 7.5, label = paste0("10
PCs"), size = 4.5, color = "blue")
>print(elbow_plot) #
```

33. Use the first 10 PCs for clustering with 'FindNeighbors()' based on the plots described in the previous section.

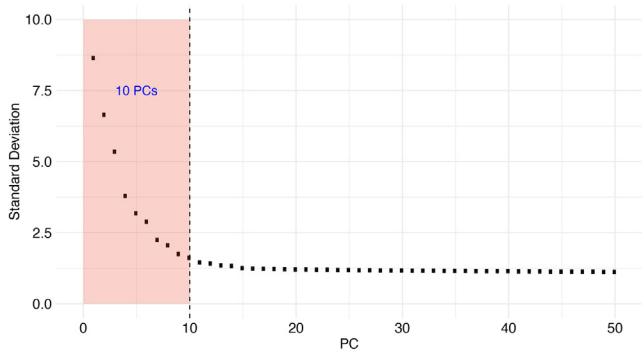


Figure 7. Representative elbow plot
Showing the relationship of the (standard) variation across the first 50 principal components.

34. Use the following values of resolution with the function 'FindClusters()': 0.20, 0.25, 0.30, 0.35, 0.40, 0.45, 0.50, 0.55, 0.60.
35. Create a UMAP using the function 'RunUMAP()' using the first 10 PCs.

Note: Resolution is one important parameter and crucial to determine the number of clusters obtained.

```
## Gt1 sample
>set.seed(1024)
>gt1_seu <- FindNeighbors(gt1_seu, dims = 1:10)
>res_2_iter <- seq(0.2,0.6, by = 0.05) # resolutions to test
# iterate over the different resolutions and return a list of Seurat objects to work with
>gt1_seu_list <- list()
>for ( res in res_2_iter ) {
  seu <- paste0("gt1_seu_", res)
  set.seed(1024)
  gt1_seu_list[[seu]] <- FindClusters(gt1_seu, resolution = res)
}
# check how many clusters by resolution
>no_clusters <- lapply(gt1_seu_list, function(x) {
length(levels(x$seurat_clusters))
})
>clust_df <- data.frame("Seurat_obj" = names(gt1_seu_list),
           "Resolution" = res_2_iter,
           "No_clusters" = unlist(no_clusters))
# iterate over the list
>gt1_seu_umap <- list()
>for ( seu in names(gt1_seu_list) ) {
```

```
set.seed(1024)
  gt1_seu_list[[seu]] <- RunUMAP(gt1_seu_list[[seu]], dims = 1:10)
  gt1_seu_umap[[seu]] <- DimPlot(gt1_seu_list[[seu]], reduction = "umap", label = TRUE) +
ggtitle(paste0("Resolution: ", clust_df[clust_df$Seurat_obj == seu, "Resolution"]))
}
## Gt2 sample
>set.seed(1024)
>gt2_seu <- FindNeighbors(gt2_seu, dims = 1:10)
>res_2_iter <- seq(0.2,0.6, by = 0.05) # resolutions to test
# iterate over the different resolutions and return a list of Seurat objects to work with
>gt2_seu_list <- list()
>for ( res in res_2_iter ) {
  seu <- paste0("gt2_seu_", res)
  set.seed(1024)
  gt2_seu_list[[seu]] <- FindClusters(gt2_seu, resolution = res)
}
# check how many clusters by resolution
>no_clusters <- lapply(gt2_seu_list, function(x) {
length(levels(x$seurat_clusters))
})
>clust_df <- data.frame("Seurat_obj" = names(gt2_seu_list),
           "Resolution" = res_2_iter,
           "No_clusters" = unlist(no__clusters))
>## iterate over the list
>gt2_seu_umap <- list()
>for ( seu in names(gt2_seu_list) ) {
  set.seed(1024)
  gt2_seu_list[[seu]] <- RunUMAP(gt2_seu_list[[seu]], dims = 1:10)
  gt2_seu_umap[[seu]] <- DimPlot(gt2_seu_list[[seu]], reduction = "umap", label = TRUE) +
ggtitle(paste0("Resolution: ", clust_df[clust_df$Seurat_obj == seu, "Resolution"]))
}
# retrieve the seurat object to work with
>gt1_seu <- gt1_seu_list$gt1_seu_0.35
>gt2_seu <- gt2_seu_list$gt2_seu_0.4
```

36. Select the resolution values, e.g., 0.35 and 0.4 for gt1 and gt2, respectively, and the corresponding clustering are used downstream.

Note: At this stage, the processed samples can be submitted to a doublet identification algorithm, such as DoubletFinder,[24] in order to remove potential doublets (for instructions see the documentation at: https://github.com/chris-mcginnis-ucsf/DoubletFinder).[24] As the identification of doublets generated from transcriptionally similar cell subtypes - homotypic doublets - is challenging, this step should be applied with caution depending on the type of data being analyzed, particularly, if it includes multicellular stages as found with Plasmodium. In this protocol, a more conservative approach is followed and, thus, doublet identification and removal is not performed at the cost of keeping some potential doublets.

37. Apply the function 'FindAllMarkers()' to find positive and negative markers for all with the following options: minimum percentage of cells that a gene needs to be expressed ('min.pct = 0.25') and a log2 fold change threshold of 0.25 ('logfc.threshold = 0.25'). The same function is used for positive markers ('only.pos = TRUE').

```
## Gt1 sample
# retrieve all the markers
>set.seed(1024)
>gt1_markers_all <- FindAllMarkers(gt1_seu, min.pct = 0.25,
logfc.threshold = 0.25)
# retrieve only positive markers
>set.seed(1024)
>gt1_markers_pos <- FindAllMarkers(gt1_seu, only.pos = TRUE,
min.pct = 0.25, logfc.threshold = 0.25)
## Gt2 sample
# retrieve all the markers
>set.seed(1024)
>gt2_markers_all <- FindAllMarkers(gt2_seu, min.pct = 0.25,
logfc.threshold = 0.25)
# retrieve only positive markers
>set.seed(1024)
>gt2_markers_pos <- FindAllMarkers(gt2_seu, only.pos = TRUE,
min.pct = 0.25, logfc.threshold = 0.25)
```

38. Export the main R objects to continue the analyses below.

```
## Gt1 sample
# create dir
>r_objs_folder <- "../results/gt1/R_objects"
>if( ! dir.exists(r_objs_folder) ) dir.create(r_objs_folder)
# save R objects
```

```
>saveRDS(object = gt1_seu, file = paste(r_objs_folder,

"gt1_seu.rds", sep = "/"))

## Gt2 sample

# create dir

>r_objs_folder <- "../results/gt2/R_objects"

>if( ! dir.exists(r_objs_folder) ) dir.create(r_objs_folder)

# save R objects

>saveRDS(object = cre3_seu, file = paste(r_objs_folder, "cre3_seu.rds", sep = "/"))
```

Note: To integrate, cluster, perform dimensional reduction, DEG and functional enrichment the next section continues by importing the processed data to integrate the two samples in order to harmonize/align shared cell types across conditions. The result is an integrated data set which is then used to identify and visualize cell populations through unsupervised clustering and projection onto the low-dimensional space, respectively. Finally, DGE between cell populations and across conditions is performed to identify conserved cluster markers and genes affected by the condition, respectively. The functions enriched based on the list of differentially expressed genes found are explored. The integration method performed is the Seurat CCA (Canonical Correlation Analysis).

39. Import the objects produced above.

```
# Import packages

>library("dplyr", quietly = TRUE)

>library("Seurat", quietly = TRUE)

>library("ggplot2", quietly = TRUE)

>library("tidyr", quietly = TRUE)

# Import Seurat objects of Gt2 and Gt1

>gt2_seu <- readRDS(file = "../results/gt2/R_objects/gt2_seu.rds")

>gt1_seu <- readRDS(file = "../results/gt1/R_objects/gt1_seu.rds")

>seu_list <- list(

  "gt2" = gt2_seu,

  "gt1" = gt1_seu

)
```

40. Normalize both samples with 'NormalizeData()' and variable features determined with 'FindVariableFeatures()' using the 'vst' method ('selection.method = "vst"') with the top 750 most highly variable features.
41. Integrated features are selected with 'SelectIntegrationFeatures()'.

```
## Normalization and Feature selection independently for each data set

>params <- list(n_var_features = 750) # list of parameters to use throughout the analysis
```

```
>seu_list <- lapply(X = seu_list, FUN = function(x) { # apply norm & var
  x <- NormalizeData(x)
  x <- FindVariableFeatures(x, selection.method = "vst", nfeatures = params$n_var_features)
})
# select features that are variable across both data sets
>params[["var_features"]] <- SelectIntegrationFeatures(object.list = seu_list)
```

42. Perform integration with 'IntegrateData()' by finding shared anchors across samples with 'FindIntegrationAnchors()' using the integrated variable features from before.
43. Scale the integrated data by ('ScaleData()'), PCA ('RunPCA()') and UMAP ('RunUMAP()') (Figure 8).
44. Perform graph-based clustering with 'FindNeighbors()' using the PCA method.
45. Define clusters with 'FindClusters()' using a resolution value that yields a similar amount of clusters as in the separate analysis.

```
## Find anchors, integrate, cluster, PCA and UMAP
# find anchors
>params[["anchors"]] <- FindIntegrationAnchors(
object.list = seu_list,
anchor.features = params[["var_features"]]
)
# integrate labels
>seu <- IntegrateData(anchorset = params[["anchors"]])
# define default assay, assay integrated
>DefaultAssay(seu) <- "integrated"
# cluster, PCA and UMAP
>params[["n_pcs"]] <- 12; params[["mth"]] <- "pca"; params[["res"]] <- 0.39;
>seu <- ScaleData(seu, verbose = FALSE)
>seu <- RunPCA(seu, npcs = params$n_pcs, verbose = FALSE)
>seu <- RunUMAP(seu, reduction = params$mth, dims = 1:params$n_pcs)
>seu <- FindNeighbors(seu, reduction = params$mth, dims = 1:params$n_pcs)
>seu <- FindClusters(seu, resolution = params$res)
```

46. Find conserved markers across the two samples with the function 'FindConservedMarkers()' on the original RNA assay for each independent cluster as follows.

```
# Find conserved markers
# create folder to save conserved markers
>conserved_markers_folder <- "../results/int_28_05_21/tables/conserved_markers"
>if ( ! dir.exists(conserved_markers_folder) ) dir.create(conserved_markers_folder,
recursive = TRUE)
```

```
# change default assay for DGE

>DefaultAssay(seu) <- "RNA"

# Run conserved markers

>conserved_markers <- list() # to save results into a list

>clts <- seu@meta.data$seurat_clusters %>% levels(.) # all the clts to loop over

>for ( clt in clts ) { # loop over the clts and get conserved markers

clt_name <- paste0("clt_", clt)

  clt_no <- as.numeric(clt)

  cell_no_gt1 <- seu@meta.data %>%

    filter(orig.ident == "Gt1" & seurat_clusters == clt) %>%

    nrow(.)

  cell_no_gt2 <- seu@meta.data %>%

    filter(orig.ident == "Gt2" & seurat_clusters == clt) %>%

    nrow(.)

  if ( (cell_no_gt1 >= 3) & (cell_no_gt2 >= 3) ) {

    set.seed(1024)

conserved_markers[[clt_name]] <- FindConservedMarkers(seu, ident.1 = clt_no, grouping.
var = "orig.ident", verbose = FALSE)

  write.table(x = cbind("Gene_id" = rownames(conserved_markers[[clt_name]]), conserved_
markers[[clt_name]]), file = paste(conserved_markers_folder, paste0(clt_name, "_conserved_
markers.tsv"), sep = "/"), sep = "\t", row.names = FALSE, quote = FALSE)

}

## Join conserved markers into one table

>conserved_markers_tbl <- NULL

>for ( clt in names(conserved_markers) ) { # loop over clusters

  sub_df <- conserved_markers[[clt]]

  col_names <- colnames(sub_df)

  sub_df[,"Gene_id"] <- rownames(conserved_markers[[clt]])

  sub_df[,"Cluster"] <- clt

  col_order <- c("Cluster", "Gene_id", col_names)

  sub_df <- sub_df[,col_order]

  # check if colnames match to just rbind

  if ( is.null(conserved_markers_tbl) ) {

    conserved_markers_tbl <- sub_df

  } else {

    stopifnot( all( colnames(conserved_markers_tbl) == col_order ) )

    conserved_markers_tbl <- rbind( conserved_markers_tbl, sub_df )

  }

}
```

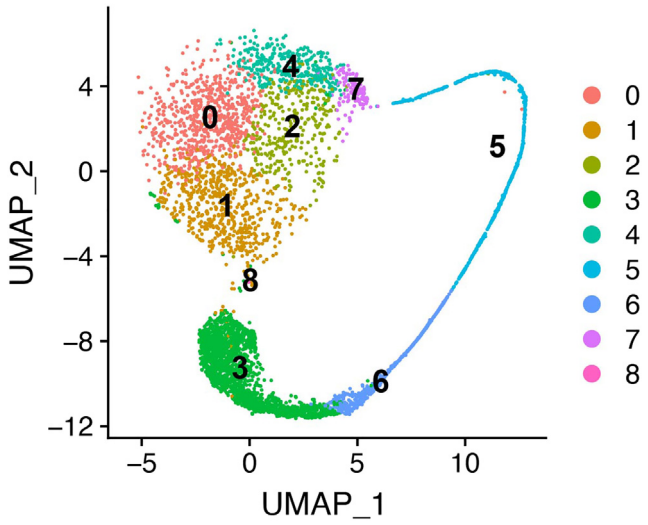


Figure 8. Representative UMAP plot of integrated *Pcc samples*
Adapted from Ramos, Ademolue et al. (2022).[1]

47. Use tables of conserved markers to define clusters identity:

Note: To understand how the transcriptomes are segregated, enquire their identity in each cluster of conserved markers in both samples. This will allow, for example, ascribing different clusters to different life cycle stages and probe for effects on *Plasmodium* life cycle progression and development. For this, we relied on the publicly available information on *Plasmodium* stage specific gene expression to enquire whether the topmost expressed genes *per* cluster were specifically expressed in a particular stage of the life cycle (Figure 9).

48. Identify the top 25 genes expressed in each cluster.

Note: We found this number to be enough to provide a defined signature of specific blood stages of *Plasmodium* development. Of note, some transcripts are expressed in most developmental stages, while others are specific for certain stages (*e.g.*, gametocytes). However, the number of genes analyzed to assure a correct ascribing of specific developmental stages to each cluster will depend on each dataset. Most of the publicly available data for *Plasmodium* stage-specific gene expression, generated from either bulk or single cell RNAseq, refers to *P. falciparum*, *P. vivax* or *P. berghei in vitro*. To analyze the stage-specificity of *Pcc* gene sets, it is required to find the orthologues in these parasite species. Use available platforms, e.g., PlasmoDB (www.plasmodb.org), to identify *P. falciparum* or *P. berghei*. orthologues of the genes of interest. Assess gene expression throughout the *Plasmodium* life cycle. Use publicly available data to enquire on stage-specificity of gene expression. We found that Malaria Cell Atlas (www.malariacellatlas.org) to provide an excellent compilation of gene expression from bulk (smatseq2) or single cell (Chromium 10×) data acquired from different life cycle stages of *Plasmodium* spp., including *P. falciparum* or *P. berghei*.

49. Confirm *Plasmodium* stage-specific gene expression for each cluster.

⚠ CRITICAL: Some clusters may have genes expressed typically in different *Plasmodium* developmental stages, indicating: i) Developmental stage transition (*e.g.*, from rings to trophozoite where ring-stage specific genes are co-expressed with metabolic genes, typically highly expressed in trophozoites); ii) Sexual commitment (*e.g.*, gene expression

signature corresponding to asexual stages overlaps with gene signatures associated with parasite sexual commitment, suggesting a transition from merozoites, rings or trophozoites into gametocytes).

50. Analyze differential gene expression (DGE) across samples for each cluster using the function 'FindMarkers()'.
51. Use the identity of the cells belonging to each one of the two samples, Gt2 or Gt1, by cluster to compare genes differentially expressed across each cluster.

Note: By default, this function uses the method Wilcoxon Rank Sum test on the normalized gene expression data.

```
## DGE: Gt2 vs Gt1

# Add metadata field to hold sample type and cluster id & change the identity to this

>stopifnot(DefaultAssay(seu) == "RNA")

>seu[["treat"]] <- paste(seu$orig.ident, Idents(seu), sep = "_")

>seu[["clusters"]] <- Idents(seu)

>Idents(seu) <- seu$treat

# create folder directory to save files

>dge_table_folder <- "../results/int_28_05_21/tables/dge_tables"

>if ( ! dir.exists(dge_table_folder) ) dir.create(dge_table_folder)

# loop over clusters & do DGE for each cluster among samples

>dge <- list()

>no_int_clts <- levels(seu$clusters)

>for ( clt in no_int_clts ) { # loop over cluster

  cell_no_gt1 <- seu@meta.data %>%

    filter(orig.ident == "Gt1" & clusters == clt) %>%

    nrow(.)

  cell_no_gt2 <- seu@meta.data %>%

    filter(orig.ident == "Gt2" & clusters == clt) %>%

    nrow(.)

  if ( (cell_no_gt2 >= 3) & (cell_no_gt1 >= 3) ) {

    clt_name <- paste0("clt_", clt) # name of the current cluster

    ctrl_cells <- paste0("Gt1_", clt) # control/reference name of the current groups of cells
to compare

    treat_cells <- paste0("Cre3_", clt) # treatment name of the current groups of cells to
compare

    set.seed(1024)

    dge[[clt_name]] <- FindMarkers(seu, ident.1 = treat_cells, ident.2 = ctrl_cells,
verbose = FALSE) # do dge

    write.table(x = cbind("Geneid" = rownames(dge[[clt_name]]), dge[[clt_name]]), file =
paste(dge_table_folder, paste0(clt_name, "_Gt2_vs_Gt1_dge_table.tsv"), sep = "/"), sep =
"\t", quote = FALSE, row.names = FALSE
```

```
    )
  }
}
# restore Seurat identity to clusters
>Idents(seu) <- seu$clusters
## Merge DGE tables in order to plot them below
>dge_all <- NULL
>for ( clt in names(dge) ) {
  sub_df <- dge[[clt]]
  col_names <- colnames(sub_df)
  sub_df[,"Geneid"] <- rownames(dge[[clt]])
  sub_df[,"Cluster"] <- clt
  col_order <- c("Cluster", "Geneid", col_names)
  sub_df <- sub_df[,col_order]
  if ( is.null(dge_all) ) {
    dge_all <- sub_df
  } else {
    stopifnot( all( colnames(sub_df) == colnames(dge_all) ) )
    dge_all <- rbind(dge_all, sub_df)
  }
}
>write.table(x = dge_all, file = paste(dge_table_folder, "Cre3_vs_Lox2_dge_table_for_all_-
clusters.tsv", sep = "/"), sep = "\t", quote = FALSE, row.names = FALSE)
```

52. Highlight the results in volcano plots and heatmaps.
53. Use gprofiler2 R package to perform functional enrichment analysis on the differentially expressed genes (i.e., up- and down-regulated) between samples by cluster (v.0.2.0[7]), an interface to the g:Profiler web browser tool.
54. Apply the function 'gost()' to perform functional enrichment analysis, based on genes up- or down-regulated, between each pairwise comparison, against the annotated genes ('domain_scope = "annotated"') of the organism *P. chabaudi* ('organism = "pchabaudi"').
55. Order gene lists by increasing adjusted p-value ('ordered_query = TRUE') to generate a GSEA (Gene Set Enrichment Analysis) style *p*-values.

    Note: This allows to start the enrichment testing from the topmost biological relevant genes with subsequent tests involving larger sets of genes. In addition, only statistically significant ('user_threshold = 0.05') enriched functions are returned ('significant = TRUE') after multiple testing corrections with the default method g:SCS ('correction_method = "g_SCS"').

56. Add evidence codes to the final result ('evcodes = TRUE'). Functional databases available to *P. chabaudi* are the following: Gene Ontology (GO or by branch GO:MF, GO:BP, GO:CC), KEGG.
57. Subject all the clusters to functional enrichment analysis.
58. Parse the DGE results for gprofiler2 as follows:

```
## Get up and down gene lists and parse them

>reg_gene_list <- list()

>for ( clt in names(dge) ) { # select dge by cluster and retrieve name

  sub_df <- dge[[clt]]

  sub_df[,"Geneid"] <- rownames(sub_df)

  gene_ids_up <- sub_df %>%

    filter(p_val_adj < 0.05 & avg_log2FC > 0) %>%

    arrange(p_val_adj) %>%

    pull(Geneid)

  gene_ids_down <- sub_df %>%

    filter(p_val_adj < 0.05 & avg_log2FC < 0) %>%

    arrange(p_val_adj) %>%

    pull(Geneid)

  reg_gene_list[[clt]] <- list()

  reg_gene_list[[clt]][["up"]] <- gsub(pattern = "-", replacement = "_", x = gene_ids_up)

  reg_gene_list[[clt]][["down"]] <- gsub(pattern = "-", replacement = "_", x = gene_ids_
down)

}

# create folders

>r_object_folder <- "../results/int_28_05_21/R_objects"

>if ( ! dir.exists(r_object_folder) ) dir.create(r_object_folder)

>func_enrich_folder <- "../results/int_28_05_21/tables/functional_enrichment"

>if ( ! dir.exists(func_enrich_folder) ) dir.create(func_enrich_folder)
```

59. Submit the list of differentially up- and down-regulated genes by cluster to g:profiler2 as follows:

```
### Functional enrichment of DEG

# Run gprofiler2

>func_enrich <- list()

>set_base_url("https://biit.cs.ut.ee/gprofiler_archive3/e102_eg49_p15")

>for ( clt in names(reg_gene_list) ){ # loop over list and do functional enrichment

  #print(get_base_url())

  func_enrich[[clt]] <- list()

  set.seed(1024)

  func_enrich[[clt]][["up"]] <- gost(query = reg_gene_list[[clt]][["up"]], organism =
"pchabaudi", ordered_query = TRUE, multi_query = FALSE, significant = TRUE, exclude_iea =
FALSE, measure_underrepresentation = FALSE, evcodes = TRUE, user_threshold = 0.05, correction_
method = "g_SCS", domain_scope = "annotated", custom_bg = NULL, numeric_ns = "", sources = NULL)

  if ( ! is.null(func_enrich[[clt]][["up"]]$result) ) {
```

```
    sub_df_up <- func_enrich[[clt]][["up"]]$result %>%
    apply(X = ., MARGIN = 2, FUN = function(x) as.character(x))
    write.table(x = sub_df_up,
         file = paste(func_enrich_folder, paste0(clt, "_up_functional_enrichment_table.tsv"),
                sep = "/"),
         quote = FALSE, sep = "\t", row.names = FALSE, col.names = TRUE)
set.seed(1024)
  func_enrich[[clt]][["down"]] <- gost(query = reg_gene_list[[clt]][["down"]], organism =
"pchabaudi", ordered_query = TRUE, multi_query = FALSE, significant = TRUE, exclude_iea =
FALSE, measure_underrepresentation = FALSE, evcodes = TRUE, user_threshold = 0.05, correction_
method = "g_SCS", domain_scope = "annotated", custom_bg = NULL, numeric_ns = "", sources = NULL)
  if ( ! is.null(func_enrich[[clt]][["down"]]$result) ) {
    sub_df_down <- func_enrich[[clt]][["down"]]$result %>%
    apply(X = ., MARGIN = 2, FUN = function(x) as.character(x))
    write.table(x = sub_df_down,
file = paste(func_enrich_folder, paste0(clt, "_down_functional_enrichment_table.tsv"),
             sep = "/"),
         quote = FALSE, sep = "\t", row.names = FALSE, col.names = TRUE)
    }
}
# Create R object
>saveRDS(object = func_enrich, file = paste(r_object_folder, "func_enrich.rds", sep = "/"))
The results were explored through tables and plots.
The main integrated object was exported.
>saveRDS(object = seu, file = paste(r_object_folder, "seu.rds", sep = "/"))
```

60. Explore the results through tables and plots.
61. Export the main integrated object.

Note: To perform RNA velocity analysis, which uses the ratio of spliced/unspliced transcripts to derive cell state transitions and infers pseudotime and cell trajectory the required inputs need to be obtained. The two input requirements are: spliced/unspliced matrices from the alignment sample files and the UMAP embedding. The result is the inference of cell state transitions projected onto UMAP space that allows to explore the cell development trajectory.

62. Perform RNA velocity analysis of scRNA-seq samples of *Pcc* with RNA by projecting the integrated UMAP obtained with Seurat analysis described above.
63. Performe RNA velocity analysis in two steps, following the guidelines from https://github.com/basilkhuder/Seurat-to-RNA-Velocity:
    a. Generate the '.loom' file with the spliced and unspliced count matrices
    b. Estimate the RNA velocity from the spliced and unspliced matrices by using the velocyto (v.0.17.17[8]) CLI (Command-Line Interface) tool and running the bash script:

```
> ./velocyto_script.sh &> velocyto_log.log
```

64. The content of the 'velocyto_script.sh' bash script is summarized below:

```
## Variables

# PATH

>AFS_PATH=/afs/igc.gulbenkian.pt/folders/UBI/PROJECTS/UBI-2021/ongoing/
2012_miguel_elisa

# GTF and bam files for Gt2 and Gt1 samples

>SAMPLES='Gt2 Gt1'

>bam_gt2=${AFS_PATH}/data/ftp01.igc.gulbenkian.pt/Gt2_count_full/outs/
possorted_genome_bam.bam

>bam_gt1=${AFS_PATH}/data/ftp01.igc.gulbenkian.pt/Gt1_count_full/outs/
possorted_genome_bam.bam

>GTF=${AFS_PATH}/data/ftp01.igc.gulbenkian.pt/Pchabaudi2/genes/genes.gtf

# activate conda first

>source /home/agsousa/miniconda3/etc/profile.d/conda.sh

>conda activate velocyto

>parallel -v \

  velocyto run10x -@ 8 --samtools-memory 5000 \

  ${AFS_PATH}/data/ftp01.igc.gulbenkian.pt/{}_count_full \

  $GTF ::: $SAMPLES >> velocyto_script.log 2>&1

>conda deactivate
```

65. Process both samples in parallel with GNU parallel (v.20161222[25]).

Note: Sample folders generated with the cellranger pipeline are provided as input, which contain the folder 'outs' with the '.bam' alignment file and the filtered barcodes. In addition, the GTF file of *Pcc* is needed.

66. Before estimating velocities, export the identity of the cells, clusters as well as the UMAP embeddings from the integrated single-cell data to project the velocities. This data is exported in R as follows:

```
## Import packages

>library("Seurat", quietly = TRUE)## Import integrated Seurat object: Gt2 and Gt1

# import obj integrated

>seu <- readRDS(file = "../results/int/R_objects/seu.rds")

# Seurat: split objects by sample

>sobjList <- SplitObject(seu, split.by = "orig.ident")

# dir to save
```

```
>seurat_data <- "../results/velocyto/seurat_data"

>if ( ! dir.exists(seurat_data) ) dir.create(seurat_data, recursive = TRUE)

# save cell_ids for both samples

>write.csv(Cells(sobjList$Gt2), file = paste(seurat_data, "gt2_samp_cellID_obs.csv", sep =
"/"), row.names = FALSE)

>write.csv(Cells(sobjList$Gt1), file = paste(seurat_data, "gt1_samp_cellID_obs.csv", sep =
"/"), row.names = FALSE)

# save UMAP embeddings for both samples

>write.csv(Embeddings(sobjList$Gt2, reduction = "umap"), file = paste(seurat_data,"gt2_
UMAP_cell_embeddings.csv", sep = "/"))

write.csv(Embeddings(sobjList$Gt1, reduction = "umap"), file = paste(seurat_data,"lox2_
UMAP_cell_embeddings.csv", sep = "/"))
```

67. Estimate RNA velocity with the stochastic model option from the spliced and unspliced matrices using the python package scvelo (v.0.2.2[9]) through jupyter notebook.
68. Project velocities onto the integrated UMAP embeddings as described below.

```
# save cluster information

>write.csv(sobjList$Gt2@meta.data[,"seurat_clusters", drop = FALSE],

     file = paste(seurat_data,"gt2_clusters.csv", sep = "/"))

>write.csv(sobjList$Gt1@meta.data[,"seurat_clusters", drop = FALSE],

     file = paste(seurat_data,"gt1_clusters.csv", sep = "/"))

# export the colors used in Seurat UMAP clusters

# Gt2

>gt2_umap_colors <- DimPlot(sobjList$Gt2)

>gt2_umap_colors <- ggplot2::ggplot_build(gt2_umap_colors)

>gt2_umap_colors <- gt2_umap_colors$data[[1]]

>gt2_umap_colors <- gt2_umap_colors[,"colour", drop = FALSE]

>gt2_umap_colors[,"cluster"] <- row.names(sobjList$Gt2@meta.data)

>write.csv(gt2_umap_colors, file = paste(seurat_data,"gt2_umap_colors.csv", sep = "/"), row.names = FALSE)

# Gt1

>gt1_umap_colors <- DimPlot(sobjList$Gt1)

>gt1_umap_colors <- ggplot2::ggplot_build(gt1_umap_colors)

>gt1_umap_colors <- gt1_umap_colors$data[[1]]

>gt1_umap_colors <- gt1_umap_colors[,"colour", drop = FALSE]

>gt1_umap_colors[,"cluster"] <- row.names(sobjList$Gt1@meta.data)

>write.csv(gt1_umap_colors, file = paste(seurat_data,"gt1_umap_colors.csv", sep = "/"), row.names = FALSE)

# Import packages

>import os
```

```
>import random

>import anndata

>import scvelo as scv

>import pandas as pd

>import numpy as np

>import matplotlib as plt

#%load_ext rpy2.ipython

# Keep reproducibility

>random.seed(1024)

## import loom files

# path to the loom files

>gt2_loom_file = "../data/ftp01.igc.gulbenkian.pt/Gt2_count_full/velocyto/Gt2_count_full.loom"

>gt1_loom_file = "../data/ftp01.igc.gulbenkian.pt/Gt1_count_full/velocyto/Gt1_count_full.loom"

>sample_gt2 = anndata.read_loom(gt2_loom_file)

>sample_gt1 = anndata.read_loom(gt1_loom_file)

## parse cell ids from loom files to match the Seurat cell ids

# gt2

>cell_ids_gt2 = sample_gt2.obs_names.to_list()

# remove the prefix 'Gt2_count_full:' and sufix 'x' from the cell ids of the loom file

# add the sufix "-1_1"

>cell_ids_gt2_parsed = [ cell[16:32] + "-1_1" for cell in cell_ids_gt2 ]

>sample_gt2.obs_names = cell_ids_gt2_parsed

# gt1

>cell_ids_gt1 = sample_gt1.obs_names.to_list()

# remove the prefix 'Gt1_count_full:' and sufix 'x' from the cell ids of the loom file

# add the sufix "-1_2"

>cell_ids_gt1_parsed = [ cell[16:32] + "-1_2" for cell in cell_ids_gt1 ]

>sample_gt1.obs_names = cell_ids_gt1_parsed

## import seurat metadata

# cell ids

>seurat_data = "../results/velocyto/seurat_data"

>sample_gt2_obs = pd.read_csv(seurat_data + "/" + "gt2_samp_cellID_obs.csv")

>sample_gt1_obs = pd.read_csv(seurat_data + "/" + "gt1_samp_cellID_obs.csv")

## import umap embeddings

>umap_cord_gt2 = pd.read_csv(seurat_data + "/" + "gt2_UMAP_cell_embeddings.csv")

>umap_cord_gt1 = pd.read_csv(seurat_data + "/" + "gt1_UMAP_cell_embeddings.csv")

## import cell_clusters

>cell_clusters_gt2 = pd.read_csv(seurat_data + "/" + "gt2_clusters.csv")
```

```
>cell_clusters_gt1 = pd.read_csv(seurat_data + "/" + "gt1_clusters.csv")

## filter loom files based on Seurat cell ids (filtered)

>sample_gt2 = sample_gt2[np.isin(sample_gt2.obs.index, sample_gt2_obs["x"])] # 15543 cells to 15537

>sample_gt1 = sample_gt1[np.isin(sample_gt1.obs.index, sample_gt1_obs["x"])] # 5204 cells to 5176

### Parse data

## rename column

# Gt2

>sample_gt2_index = pd.DataFrame(sample_gt2.obs.index)

>sample_gt2_index = sample_gt2_index.rename(columns = {0:'Cell ID'})

# Gt1

>sample_gt1_index = pd.DataFrame(sample_gt1.obs.index)

>sample_gt1_index = sample_gt1_index.rename(columns = {0:'Cell ID'})

# rename column from UMAP df

>umap_cord_gt2 = umap_cord_gt2.rename(columns = {'Unnamed: 0':'Cell ID'})

>umap_cord_gt1 = umap_cord_gt1.rename(columns = {'Unnamed: 0':'Cell ID'})

# merge UMAP and index data

>umap_ordered_gt2 = sample_gt2_index.merge(umap_cord_gt2, on = "Cell ID")

>umap_ordered_gt1 = sample_gt1_index.merge(umap_cord_gt1, on = "Cell ID")

# remove 1st 'Cell ID' column and add UMAP coordinates

# Gt2

>umap_ordered_gt2 = umap_ordered_gt2.iloc[:,1:]

>sample_gt2.obsm['X_umap'] = umap_ordered_gt2.values

# Gt1

>umap_ordered_gt1 = umap_ordered_gt1.iloc[:,1:]

>sample_gt1.obsm['X_umap'] = umap_ordered_gt1.values

## add cluster colors

# Gt2

>cell_clusters_gt2 = cell_clusters_gt2.rename(columns = {'Unnamed: 0':'Cell ID'})

>cluster_ordered_gt2 = sample_gt2_index.merge(cell_clusters_gt2, on = "Cell ID")

>cluster_ordered_gt2 = pd.Series(cluster_ordered_gt2['seurat_clusters'].apply(str).values, index = cluster_ordered_
gt2['Cell ID'])

#cluster_ordered_gt2 = cluster_ordered_gt2.iloc[:,1:]

>sample_gt2.obs['clusters'] = cluster_ordered_gt2

>sample_gt2.obs['clusters'] = sample_gt2.obs['clusters'].astype('category')

>sample_gt2.uns['Cluster_colors'] = cluster_ordered_gt2.values

# Gt1

>cell_clusters_gt1 = cell_clusters_gt1.rename(columns = {'Unnamed: 0':'Cell ID'})

>cluster_ordered_gt1 = sample_gt1_index.merge(cell_clusters_gt1, on = "Cell ID")
```

```
>cluster_ordered_gt1 = pd.Series(cluster_ordered_gt1['seurat_clusters'].apply(str).values, index = cluster_ordered_gt1['Cell ID'])

#cluster_ordered_gt1 = cluster_ordered_gt1.iloc[:,1:]

>sample_gt1.obs['clusters'] = cluster_ordered_gt1

>sample_gt1.obs['clusters'] = sample_gt1.obs['clusters'].astype('category')

>sample_gt1.uns['Cluster_colors'] = cluster_ordered_gt1.values

## RNA velocity

# define colors for both

>color = ["#F8766D", "#D39200", "#93AA00", "#00BA38", "#00C19F", "#00B9E3", "#619CFF", "#DB72FB", "#FF61C3"]

# Gt2

>scv.set_figure_params(facecolor="white", figsize=(8, 8))

>random.seed(1024)

>scv.pp.filter_and_normalize(sample_gt2)

>scv.pp.moments(sample_gt2)

>scv.tl.velocity(sample_gt2, mode = "stochastic")

>scv.tl.velocity_graph(sample_gt2)

>scv.set_figure_params(facecolor="white", figsize=(8, 8), fontsize = 10, dpi = 100, dpi_save = 300)

>scv.pl.velocity_embedding(sample_gt2, basis = 'umap', arrow_size = 30, arrow_length = 3, save = "gt2_RNA_velocity_cells_UMAP.svg")

>scv.pl.velocity_embedding_stream(sample_gt2, layer=['velocity'], palette = color, size = 30, save = 'gt2_RNA_velocity_stream_UMAP.svg')

# Gt1

>scv.set_figure_params(facecolor="white", figsize=(8, 8))

>random.seed(1024)

>scv.pp.filter_and_normalize(sample_gt1)

>scv.pp.moments(sample_gt1)

>scv.tl.velocity(sample_gt1, mode = "stochastic")

>scv.tl.velocity_graph(sample_gt1)

>scv.set_figure_params(facecolor="white", figsize=(8, 8), fontsize = 10, dpi = 100, dpi_save = 300)

>scv.pl.velocity_embedding(sample_gt1, basis = 'umap', arrow_size = 30, arrow_length = 3, save = "gt1_RNA_velocity_cells_UMAP.svg")

>scv.pl.velocity_embedding_stream(sample_gt1, layer=['velocity'], size = 30, palette = color,save = 'lox2_RNA_velocity_stream_UMAP.svg')
```

EXPECTED OUTCOMES

Around 2 × 10^6 iRBCs should be obtained from 3 × 10^6 sorted iRBC after the "washing steps". This yield is highly dependent on the method used for discarding the supernatant during the washing steps and the extent of hemolysis imposed by the procedure.

The single-cell sequencing approach enables to discriminate life-cycle specific parasite stages during the blood stage of infection. Taking into consideration that *Pcc* exhibits a circadian regulated, synchronous life cycle, specific parasite stages are more prevalent depending on the time of the

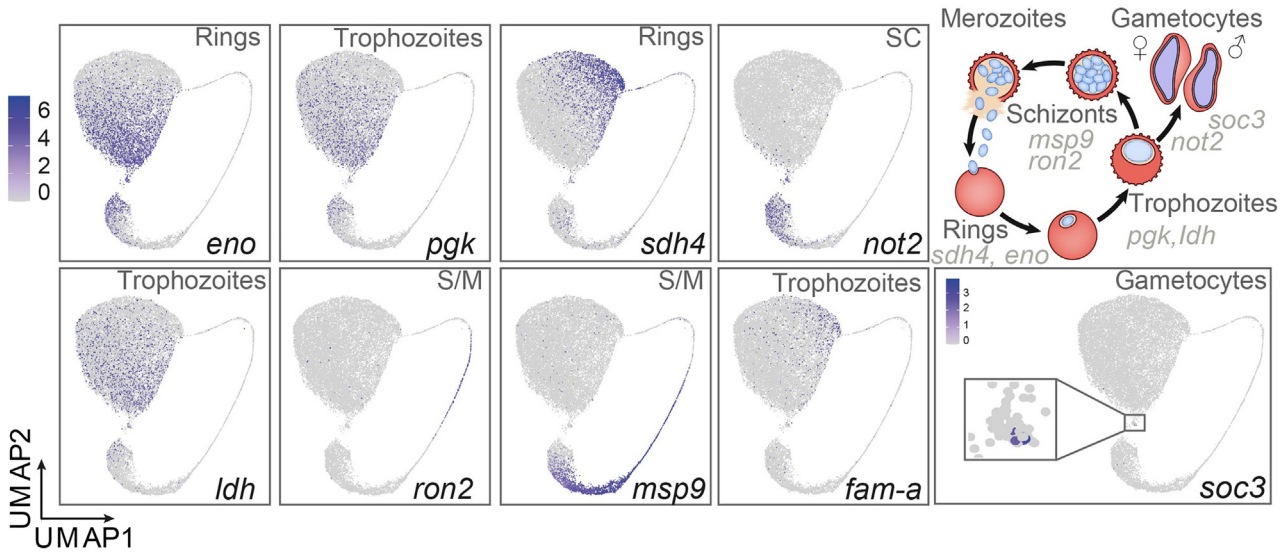


Figure 9. UMAP plots showing *Pcc* stage-specific gene expression

Based on the expression pattern different clusters are assigned to specific stages of *Pcc*. Figure reprinted with permission from Ramos, Ademolue et al. (2022).[1]

sample collection. Namely, ring stages are the main developmental stage observed during the morning and early afternoon, while trophozoites start developing throughout late afternoon and evening. During the night, mature trophozoite stages progress into schizont stages, which reach full maturity and burst, releasing merozoites into circulation, during the night. Gametocytes, however, do not follow this circadian regulation and last a few days in circulation, being observed, when present, at any time of day. Yet, it should be possible to identify transcriptional "fingerprints" of all blood *Pcc* life-cycle stages. Depending on extrinsic factors (e.g., host genotype, ambient temperature) it is possible to detect differentiation towards the sexually committed parasite stages (i.e., gametocytes), allowing to identify how host metabolic and/or immune responses regulate the progression of blood stages of *Plasmodium* through different life-cycle stages parasite and how transition among these life-cycle stages occurs, as assessed by pseudo-time analysis. This allows estimating whether the parasite follows the normal transition among different asexual life cycle stages, or whether the life-cycle of parasite is arrested at specific a stage and/or whether it engages into gametogenesis, to produce circulating gametocytes that can be transmitted through a mosquito bite.

As different life-cycle parasite stages have specific transcriptional programs, single cell analysis allows to compare transcriptional regulation at specific parasite stages. Comparing life-cycle specific parasite stages from different host genotypes also allows to identify the impact of the host metabolic and/or immune responses on parasite transcriptional regulation.

LIMITATIONS

Asexual stages of *Plasmodium* parasites replicate in RBC and depending on the time of blood collection, a single RBC might contain several parasites. This can also occur in the event of a single RBC being infected by multiple parasites (*i.e.,* merozoites). The 10× Single cell technology does not allow to distinguish whether one or several parasites are present in a single iRBC.

Using a mouse adapted GFP-tagged *Pcc* facilitates sorting of circulating iRBCs for scRNA-Seq analyzes. In contrast to *P. berghei* or *P. falciparum* however, *Pcc* transcripts lack, to a large extent, detailed functional analysis. To associate UMAP clusters, obtained by scRNA-Seq analysis, with

different asexual stages of *Pcc* it is necessary therefore to identify stage-specific transcripts. Since those transcripts are largely unknown for *Pcc*, identification of stage-specific transcripts can be achieved by homology analysis with other *Plasmodium* spp. (i.e., *P. berghei* or *P. falciparum*).

Bioinformatic analyses were performed using scripts or notebooks, depending on the suitability of the task, software used, and software versions provided to document and reproduce the data generated. Ideally, this could have been performed in a containerized environment such as docker or singularity to ensure reproducibility of the results generated and ease distribution of the specific software used as well as their versions.

TROUBLESHOOTING

Problem 1

No countable parasite under the microscope (related to step 1).

Potential solution

- If methanol is not freshly prepared, blood smears might not be fixed efficiently and thus the staining does not work. It is advisable therefore to take fresh methanol for every fixation.
- The water pH is not well adjusted, resulting in an inefficient staining of the parasite.
- A lot of debris on the slide, resulting in indistinguishable parasites from debris. This might be a result of an "old" Giemsa staining solution or insufficient filtering.
- Parasitemia is very low, resulting in scattered iRBCs. This can result from Pcc passage and/or-mouse genotype, conditions of the mouse facility (i.e., temperature changes) or the injection of too few parasites.

Problem 2

Disease severity is too high/low (related to step 1).

Potential solution

- The passage of the *Pcc* is too high/low which modulates its virulence.[15]
- Genetic background of the infected mouse affects disease progression[26] , it is advisable to run pilot experiments if genetic backgrounds, other than C57BL6/J are used, to assess the disease progression.
- Parasitemia and disease severity are closely connected. Accordingly, as for the parasitemia, the environmental conditions influence the disease progression.

Problem 3

Low iRBC yield (related to step 7).

Potential solution

- Consider that this approach must accommodate some loss of iRBC during the washing steps to allow for 1.5 x 10^6 iRBCs to be sorted. To this aim sort more than 1.5 x 10^6 iRBCs.
- Hemolysis may account for significant loss of iRBC. To limit the extent of hemolysis associated with this procedure, reduce pipetting steps to a strict minimum and, if necessary, pipette very gently. The extent of hemolysis can be qualitatively (visually) assessed by the emergence of red coloration after centrifugation. Ideally, the supernatant should be clear. If, however, the supernatant has a red color this indicates hemolysis and presumably explains low iRBC yield.
- If iRBC are lost during the washing procedure that involves the use of a vacuum pump, invert the tubes containing iRBC to discard supernatant, instead.

Problem 4

RNAseq data quality is low (related to step 17).

Potential solution

- Too many cells were loaded on the chip. This will increase the probability of doublets and clotting of cells. Make sure that while counting, the cells are well resuspended and count accurately.
- There might be a lot of ambient RNA due to cell lysis or insufficient washing of the sample. Treat the sample carefully and increase the washing steps if necessary.

Problem 5

Artificial, contaminated or excluded cell populations in scRNA-seq data (related to step 28).

Potential solution

- The strictness or conservativeness of the filtering thresholds applied, the cluster granularity chosen and the removal or not of doublets may lead to a different number of identified cell populations. Depending on the combinatorial set of data analysis steps made, the analysis can produce artificial or contaminated cell populations or exclude true cell populations. A more conservative data analysis can lead to the identification of artificial or contaminated cell populations. Artificial cell populations are easy to identify and exclude, as they often express a set of marker genes that are not biologically meaningful. In doubt, those marker genes can be confirmed by qRT- PCR.
- Contaminated cell populations mask the biological signal of the populations that they belong to and they are more difficult to identify, in part due to the sparsity nature of scRNA-seq data. Iterative rounds of analysis with increasing strictness of filtering thresholds and/or cluster granularity can help to identify and exclude contaminated cells from cell populations.
- Stricter filtering criteria can lead to the removal of true cell populations often underrepresented in the data. Using a conservative approach at the cost of including some noisy cells might be preferable over discarding potentially rare and biologically meaningful cell populations. The cell populations identified through scRNA-seq should be supported by other types of data such as microscopic imaging, particularly, if new cell populations were identified and described.

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources and reagents should be directed to and will be fulfilled by the lead contact, Miguel P. Soares (mpsoares@igc.gulbenkian.pt).

Materials availability

This study did not generate new unique reagents.

Data and code availability

Original data is available in (Ramos et al., 2022).[1] This study did not generate a unique code. All scripts used can be accessed in GitHub:https://eur04.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgithub.com%2Finflammationlab%2FscRNAseq-malaria-STAR-protocol&data=05%7C01%7Cejentho%40igc.gulbenkian.pt%7Ceca8f01c904e44cb5e9e08db2ed6d06b%7Cc0d2816d45ea47aa80f5552d7c205098%7C0%7C0%7C638155272382875673%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=YQogUqWzlOBikAtGCk6QX0BGDeO3hEQVruKiN8DG5%2BY%3D&reserved=0 or Zenodo: https://doi.org/10.5281/zenodo.8141692.

ACKNOWLEDGMENTS

We thank Dr. Joanne Thompson (University of Edinburgh) for *Pcc* AS-GFPML parasites and excellent support from IGC's Advanced Imaging, Flow Cytometry & Antibody, and Genomics facilities. S.R.

was supported by Fundação Para a Ciência e Tecnologia (FCT; 5723/2014; FEDER029411), by the Gulbenkian foundation (IBB2017) to T.W.A., and by the Deutsche Forschungsgemeinschaft ("Balance of the Microverse" DFG, EXC 2051; 390713860) to E.J. The M.P.S. laboratory is supported by the Gulbenkian, "La Caixa" (HR18-00502) and FCT (5723/2014; FEDER029411; FEDER/29411/2017 – GlucoInfect; PTDC/MED-FSL/4681/2020 – Infectenergy; 2022.02426.PTDC – MalBil); by the Oeiras-ERC Frontier Research Incentive Awards, SymbNET Research Grants (H2020-WIDESPREAD-2020-5-952537); and by Congento (LISBOA-01-0145-FEDER-022170). M.P.S. is an associate member of the Deutsche Forschungsgemeinschaft ("Balance of the Microverse" DFG, EXC 2051; 390713860).

AUTHOR CONTRIBUTIONS

Methodology, E.J., A.G.G.S., S.R., T.W.A., J.S., J.C., D.B.; investigation, S.R., T.W.A., E.J.; writing – original draft, E.J., A.G.G.S., S.R., J.S., J.C., M.M.; writing – review & editing, R.B.L., J.J., M.P.S.

DECLARATION OF INTERESTS

The authors declare no competing interests.

REFERENCES

1. Ramos, S., Ademolue, T.W., Jentho, E., Wu, Q., Guerra, J., Martins, R., Pires, G., Weis, S., Carlos, A.R., Mahú, I., et al. (2022). A hypometabolic defense strategy against malaria. Cell Metab. *34*, 1183–1200.e12. https://doi.org/10.1016/j.cmet.2022.06.011.

2. Marr, E.J., Milne, R.M., Anar, B., Girling, G., Schwach, F., Mooney, J.P., Nahrendorf, W., Spence, P.J., Cunningham, D., Baker, D.A., et al. (2020). An enhanced toolkit for the generation of knockout and marker-free fluorescent Plasmodium chabaudi. Wellcome Open Res. *5*, 71. https://doi.org/10.12688/wellcomeopenres.15587.2.

3. Donovan, J., and Brown, P. (2006). Blood collection. Curr Protoc Immunol *Chapter*. Curr. Protoc. Immunol. *Chapter 1*, 1.7.1–1.7.9. https://doi.org/10.1002/0471142735.im0107s73.

4. Dobin, A., Davis, C.A., Schlesinger, F., Drenkow, J., Zaleski, C., Jha, S., Batut, P., Chaisson, M., and Gingeras, T.R. (2013). STAR: ultrafast universal RNA-seq aligner. Bioinformatics *29*, 15–21. https://doi.org/10.1093/bioinformatics/bts635.

5. Wickham, H. (2016). ggplot2: Elegant Graphics for Data Analysis (New York: Springer-Verlag).

6. Gu, Z., Eils, R., and Schlesner, M. (2016). Complex heatmaps reveal patterns and correlations in multidimensional genomic data. Bioinformatics *32*, 2847–2849. https://doi.org/10.1093/bioinformatics/btw313.

7. Kolberg, L., Raudvere, U., Kuzmin, I., Vilo, J., and Peterson, H. (2020). gprofiler2 – an R Package for Gene List Functional Enrichment Analysis and Namespace Conversion Toolset g:Profiler. F1000Res *9*. https://doi.org/10.12688/f1000research.24956.2.

8. Hao, Y., Hao, S., Andersen-Nissen, E., Mauck, W.M., 3rd, Zheng, S., Butler, A., Lee, M.J., Wilk, A.J., Darby, C., Zager, M., et al. (2021). Integrated analysis of multimodal single-cell data. Cell *184*, 3573–3587.e29. https://doi.org/10.1016/j.cell.2021.04.048.

9. La Manno, G., Soldatov, R., Zeisel, A., Braun, E., Hochgerner, H., Petukhov, V., Lidschreiber, K., Kastriti, M.E., Lönnerberg, P., Furlan, A., et al. (2018). RNA velocity of single cells. Nature *560*, 494–498. https://doi.org/10.1038/s41586-018-0414-6.

10. Bergen, V., Lange, M., Peidli, S., Wolf, F.A., and Theis, F.J. (2020). Generalizing RNA velocity to transient cell states through dynamical modeling. Nat. Biotechnol. *38*, 1408–1414. https://doi.org/10.1038/s41587-020-0591-3.

11. Team, R.C. (2014). R: A language and environment for statistical computing. MSOR connections *1*.

12. Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., et al. (2020). Array programming with NumPy. Nature *585*, 357–362. https://doi.org/10.1038/s41586-020-2649-2.

13. Hunter, J.D. (2007). Matplotlib: A 2D Graphics Environment. Comput. Sci. Eng. *9*, 90–95. https://doi.org/10.1109/mcse.2007.55.

14. Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., et al. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat. Methods *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2.

15. Spence, P.J. (2013). Vector transmission regulates immune control of Plasmodium virulence. Nature *498*, 228–231.

16. Castillo, M.R., Hochstetler, K.J., Greene, D.M., Firmin, S.I., Tavernier, R.J., Raap, D.K., and Bult-Ito, A. (2005). Circadian rhythm of core body temperature in two laboratory mouse lines. Physiol. Behav. *86*, 538–545. https://doi.org/10.1016/j.physbeh.2005.08.018.

17. Kalsbeek, A., la Fleur, S., and Fliers, E. (2014). Circadian control of glucose metabolism. Mol. Metab. *3*, 372–383. https://doi.org/10.1016/j.molmet.2014.03.002.

18. Davis, N.M., Lissner, M.M., Richards, C.L., Chevée, V., Gupta, A.S., Gherardini, F.C., and Schneider, D.S. (2021). Metabolomic Analysis of Diverse Mice Reveals Hepatic Arginase-1 as Source of Plasma Arginase in Plasmodium chabaudi Infection. mBio *12*, e0242421. https://doi.org/10.1128/mBio.02424-21.

19. Xie, Y., Allaire, J.j., Grolemund, G., and Ebscohost. (2018). R Markdown : The Definitive Guide, 1st Edition (Chapman and Hall/CRC).

20. Xie, Y., Dervieux, C., and Riederer, E. (2020). R Markdown Cookbook, 1st Edition (Chapman & Hall/CRC).

21. Luecken, M.D., and Theis, F.J. (2019). Current best practices in single-cell RNA-seq analysis: a tutorial. Mol. Syst. Biol. *15*, e8746. https://doi.org/10.15252/msb.20188746.

22. Real, E., Howick, V.M., Dahalan, F.A., Witmer, K., Cudini, J., Andradi-Brown, C., Blight, J., Davidson, M.S., Dogga, S.K., Reid, A.J., et al. (2021). A single-cell atlas of Plasmodium falciparum transmission through the mosquito. Nat. Commun. *12*, 3196. https://doi.org/10.1038/s41467-021-23434-z.

23. Marguerat, S., and Bähler, J. (2012). Coordinating genome expression with cell size. Trends Genet. *28*, 560–565. https://doi.org/10.1016/j.tig.2012.07.003.

24. McGinnis, C.S., Murrow, L.M., and Gartner, Z.J. (2019). DoubletFinder: Doublet Detection in Single-Cell RNA Sequencing Data Using Artificial Nearest Neighbors. Cell Syst. *8*, 329–337.e4. https://doi.org/10.1016/j.cels.2019.03.003.

25. Tange, O. (2011). Gnu parallel-the command-line power tool.; login. The USENIX Magazine *36*, 42–47.

26. Davis, N.M. (2021). Metabolomic Analysis of Diverse Mice Reveals Hepatic Arginase-1 as Source of Plasma Arginase in Plasmodium chabaudi Infection. Mbio *12*, e02424–21.