
Galloway Lab Protocols

Galloway Lab

Jan 13, 2021

CONTENTS

1	Protocols	3
1.1	Analytical protocols	3
1.2	Cloning protocols	12
1.3	Protein production	17
1.4	TC protocols	27
2	Recipes	41
2.1	Bacterial recipes	41
2.2	TC recipes	42
3	Tech documentation	47
3.1	Repository setup	47
4	Bootcamp	49
4.1	IAP Bootcamp	49
5	Contributor guide	89
5.1	Local environment setup	89
5.2	Local building	93
5.3	Online editing through Github	93
5.4	Repository layout	95
5.5	Basics of reStructuredText	96
6	Contributing	107

This is the home of all of the Galloway Lab protocols!

Click [here](#) to access the PDF version.

PROTOCOLS

1.1 Analytical protocols

1.1.1 ATAC-seq

Transposase buffer preparation

Estimated time: 1.5 hours.

1. Order the following primers:

Primer name	Scale	Purification	5' modification	Sequence
Tn5ME_rev	25nmol	DSL	Phosphate	<i>CTGTCTCTTATACACATCT</i>
Tn5ME_A	50nmol	HPLC	Cy5	<i>TCGTCGGCAGCGTCAGATGTG-TATAAGAGACAG</i>
Tn5ME_B	50nmol	HPLC	Cy5	<i>GTCTCGTGGGCTCGGAGATGTG-TATAAGAGACAG</i>

2. Resuspend each primer separately to 100 uM (the standard stock dilution, nmol * 10 uL).
3. Prepare 1:1 solutions of Tn5ME_rev / Tn5ME_A (A+rev) and Tn5ME_rev / Tn5ME_B (B+rev). Each primer is now at 50 uM.
4. Assemble these double stranded oligos by heating to 95C for 5 minutes, followed by slow cooling to room temperature. To do this on a thermocycler, use a program that does ends after the denaturation step, instead of ending in a hold step.
5. Prepare 5 uM stock solution of Tn5/Cy5. Prepare this either from freshly purified Tn5 in dialysis buffer:

Component	Volume fraction
A+rev primers	0.125
B+rev primers	0.125
Glycerol	0.4
Dialysis buffer	0.12
50 uM Tn5	0.1
DI H2O	0.13

If Tn5 is already stored in a 50% glycerol/50% 2x dialysis stock solution at 20 uM, use the following recipe:

Component	Volume fraction
A+rev primers	0.125
B+rev primers	0.125
Glycerol	0.25
20 uM Tn5 + Glycerol	0.25
DI H2O	0.25

ATAC-seq

We need the following prepared buffers:

- **Nuclear permeabilization buffer:**

Component	Concentration	Amount/100 mL final
Tris-Cl	10 mM	0.1576 g
NaCl	10 mM	0.058 g
MgCl ₂ (anhydrous)	3 mM	0.0287 g
Igepal CA-630	0.01%	10 uL
HCl	to pH 7.4	

- **Wash buffer:**

Component	Concentration	Amount/100 mL final
PBS	Base	
SDS	0.01%	.01 g
EDTA	50 mM	1.461 g

- **2x TD buffer** (from [this nature paper](https://www.nature.com/articles/nprot.2013.118)¹):

¹ <https://www.nature.com/articles/nprot.2013.118>

Component	Concentration	Amount/L final
Tris	20 mM	3.264 g
MgCl ₂	10 mM	0.95 g
DI H ₂ O	main solvent	
Acetic acid	to pH 7.6, before DMF addition	
Dimethylformamide(DMF)	20% (v/v)	

1. After cell fixation, permeabilize cells with lysis buffer for 10 minutes at room temperature.
2. Wash with PBS twice.
3. Prepare transpose mixture:

Component	Volume fraction
2x TD buffer	0.5
5 mM assembled Tn5	0.02
DI H ₂ O	0.48

4. Add 50 μ L of transpose mixture to cells to be transposed. Place the cells in a humid 37C box for 30 minutes.
5. For plated cells, wash with wash buffer three times, for 15 minutes each at 55C. For suspended cells, wash twice.
6. Add PBS media to cells and image/flow the resulting cells.

1.1.2 Attune operation

Startup

Note: Purpose: To make sure the lines are filled.

Run enhanced start up if > 3 days since the last run. Enhanced startup is just running “Startup” protocol twice. Then run performance test. If an instrument button is grayed out, try hitting refresh.

1. Remove all plates from the Cytkick.
2. Check autosample bottle connections
3. Focusing fluid has single “click” lock.
4. Important: Waste bottle (black) requires a double “click” for lock
5. Note: Connectors have disconnect “buttons” to release the connection.
6. Important: Bottle sensors need to be plugged in for performance. Do NOT remove by pulling on line, remove by pulling on solid head of sensor connector. And do NOT force.
7. Check the fluids inside the Attune.
8. To fill the Focus Fluid, slide the bottle out and add fluid. Or remove entirely. Bottle must be dry to avoid trigger leak sensor.
9. Run Debubble. Add 3 mL of Debubble Solution to tube and run twice.
10. Performance test run daily but make sure to use original filter settings (not FP special filters)

Cleaning

Note: Deep cleaning recommended once a week or every two weeks. Run with Hellmanex Solution.

Hellmanex can be used for deep cleaning. It is alkaline and viscous. To make Hellmanex Solution, dilute Hellmanex 1 mL into 2 mL DI water. Hellmanex Solution can be used in place of bleach for deep clean. Recommended for deep clean + shutdown. Requires a lot of subsequent flushing.

Performance Check

Important: Before running a performance check, make sure to check the filters. If a red stripe filter is in the instrument (not the side holder), it needs to be replaced with the correct filter or the performance check will not pass. Verify lot # of performance beads. Lot # change every ~4 years and require a new set of download files for proper calibration. Originally set September 11, 2020.

1. Add 3 full drops of performance beads and dilute with 2 mL of focusing fluid (or PBS).
2. After performance test, wash out bead with “Sip sanitize” and then wash solution.

Delta PMT	Status
> 10	Run Cleaning
> 20	Indicates an issue
> 50	Performance will fail

Maintenance

Important: Maintenance required every 3-6 months

1. Check syringe (side of main attune instrument behind plastic panel) for build up of salt. May need to be replaced every 6 months. If salt accumulates, this can block sliding of syringe and preventing loading of sample. Part can be ordered and easily replaced by the lab, no need for technician. Syringe should be finger tight. Do not over tighten.
2. Sip- The input straw can be bent. If bent, get part from Maintenance Kit, unscrew and replace.
3. Focus Filters (2) should be changed ever 3-6 months. When switching out, wash with 10 mL Focus Fluid in a syringe to flush filter.
4. Run Sanitize every 3-6 months.

1.1.3 Single cell qPCR

Single-cell acquisition

Estimated time: 3-5 minutes per cell.

Important: All work should be performed with RNase free reagents!

1. Cells should be plated on the lids of electrophysiology dishes to prevent interferences of dish sides with micromanipulator angle of approach.
 2. Label PCR tube strips with numbers to correspond to the number of the cell picked.
 3. Add 5 ul of Cells Direct 2X rxn (Cells Direct One-step qRT-PCR kit , Life, Cat# 11753-100) mix to each PCR tube.
 4. Pull glass micropipettes with the machine in the Chow lab. Pipettes pulled with 1st T=61.2 °C and 2nd T=47 °C is most optimal.
-

TODO

Specify glass micropipette diameter and describe machine used in this step.

5. Obtain dry ice in a cooler box for immediate freezing of cell following acquisition.
 6. Setup fluorescence on an inverted microscope, imaging at 40X.
 7. Identify isolated GFP+ cells with neuronal morphology using bright field and FITC.
 8. Use RNase inhibitor (~3 ul per 30 cells picked) to be placed in each glass micropipette prior to use in single-cell picking.
 9. Place pipette in micromanipulator. Position so that the tip comes into the field of view. Careful not to break the tip by lowering the Z position too far.
 10. Carefully move the tip to the edge of the cell and aspirate, scooping the cell off the dish and retrieving as much of the cell body and dendrites as possible.
 11. Crush pipette tip into 5 ul of 2X Rxn mix to capture cell in solution.
 12. Immediately place on dry ice to freeze. Keep on dry ice until ready to store in -80 °C freezer.
-

Note: Cells dissociate from plate more easily with greater amount of time out of incubator. Average time to acquire a single cell runs from 3-5 minutes.

Lysis, reverse transcription (RT), and specific target amplification (STA)

1. Prepare primer mix

- Each primer stock should be 200 μ M. (e.g. If primer is 40.4 nM, add 202 μ L of RNase free water (DEPC treated or other).
- For bookkeeping it is helpful to label each primer with a letter and keep in order (e.g A-Z then AA- AZ).
- Make 20 μ M stock of forward and reverse primer together (e. g “A F+R” for gene A.)
For 250 μ L F+R mix, add 25 μ L F and 25 μ L R and 200 μ L of RNase free water .
- For 200 nM (each primer at 200 nM) RT and STA primer mix of 1 mL, add 10 μ L of each 20 μ M F+R mix and bring to 1000 μ L. For 48 genes (480 μ L of primer), add 520 μ L RNase free water.

Estimated time: 1 hour for steps 2 through 7

- Remove cells from -80°C and immediately place on ice.
- Move quickly to 50 oC water bath for 15 sec defrost, then place back on ice.
- Make RT/STA mix:

Component	Volume/ 1rxn	Volume / 100 rxn
Superscript III/Platinum Taq	0.2 μ L	20 μ L
200 μ M Primer Mix	2.5 μ L	250 μ L
RNase-free water	1.3 μ L	130 μ L
<i>Total</i>	<i>4 μL</i>	<i>400 μL</i>

- Add 4 μ L of RT/STA mix per tube.
- Spin down samples in minifuge to collect all liquid.
- Place in thermocycler.
- Run thermocycler program (1 hr and 55 min)

#	Time	Temperature (°C)	Name
1	15 min	50	Reverse transcription
2	2 min	95	Initial denature
3	•	Cycle 4&5 18 times	Cycle
4	15 sec	95	Denature
5	4 min	60	Annealing and elongation
6	Hold	4	Chill, end

- Spin down samples in minifuge to collect all liquid. Steps i- n = 1 hr and 40 min)

10. Make Exonuclease I and SAP (Shrimp Alkaline Phosphatase) mix:

Component	Volume/ 1rxn	Volume / 100 rxn
Exonuclease I	0.5 uL	50 uL
SAP	1.0 uL	100 uL
RNAse-free water	2.5 uL	250 uL
<i>Total</i>	<i>4 uL</i>	<i>400 uL</i>

11. Add 4 ul of Exonuclease I SAP mix per tube.
12. Place in thermocycler, running the following program:

#	Time	Temperature (°C)	Name
1	20 min	37	Digest primers, dNTPs
2	15 min	80	Denature
3	Hold	4	Chill, end

13. Store @ -20 oC.

Single-cell qPCR on Fluidigm Biomark

Estimated time: 2 hours for steps 1 through 8

1. Defrost samples on ice.
2. Dilute 5x by adding 52 ul of H₂O to 13 ul of sample to make 65 ul of single-cell amplified cDNA (13 ul is standard volume size following above procedure).
3. Make Sso Fast Evagreen super mix (Bio-rad) with 20X DNA binding dye (Fluidigm, green cap):

Component	Volume/ 1rxn	Volume / 96 rxn
Sso Fast Evagreen super mix, low ROX	2.5 uL	480 uL
20X DNA binding dye loading reagent	0.25 uL	48 uL
RNAse-free water	2.75 uL	264 uL
<i>Total</i>	<i>5.5 uL</i>	<i>528 uL</i>

TODO

Check if the above mixture is diluted with water as written here

- i. ...2.5 uL/ rxn...96 rxn+ over= 480 uL
- ii. 20X DNA binding dye sample loading reagent. . . 0.25 uL/rxn. . . 96 rxn+ over= 48 uL

- iii. Total.....5.5uL/rxn.....96 rxn+ over
=528 uL
4. Mix samples in 96-well qPCR plate. Place samples in well corresponding to number to order correctly (e.g. Sample 1= A1, Sample 2= A2... Sample 13= B1, Sample 14= B2... Sample 25= C1 etc).
- i. Sso Fast/DNA binding Dye mix.....5.5uL/well
- ii. Single-cell STA cDNA samples.....4.5 ul/well
5. Cover with qPCR sticky lid. Vortex lightly to mix. Spin down in plate spinner and place on ice.
6. Prepare assays with 2X loading reagent (Fluidigm), F+R primers, and water. Place samples in well corresponding to number to order correctly (e.g. Assay 1 (Gene A)= A1, Assay 2 (Gene B)= A2... etc).
7. ?
- i. 2X Assay Loading reagent2.5 uL/ rxn.....4 rxn= 10 uL
- ii. 20 uM F+R for each assay.....0.25 uL/rxn.....4 rxn= 5 uL
- iii. RNase-free water.....2.25 uL/rxn.....4 rxn=5 uL
8. Cover assay lids with qPCR sticky lid. Vortex lightly to mix. Spin down in plate spinner and place on ice.
9. PRIME Chip: Add fluid to accumulators by depressing springs and O-rings and injecting control line fluid. Tilting the plate will ensure the fluid goes into the accumulators. Run "Prime" script on Fluidigm Mixer (HX for 96.96 and MX for 48.48.)

Consult Chip Pipetting Map (PN 68000130, REV B) to optimize efficiency of set-up for pipetting with multi-

(ref: http://openwetware.org/images/5/52/Fluidigm_96.96_RT_PCR_Quick_Reference_.pdf)

10. Use electronic multi-channel pipette to add 5 uL to each well.
11. Note: Special thin tips are to be used on this device compatible with loading 384-well plates. Using the 96-well plate spacing for tips more compatible with 96 well plates assays and sample preparation and requires using the Chip pipetting map to quickly and correctly load the assays and sample inlets. It is recommended to aspirate 5.2 uL and dispense 5 uL to avoid introducing air bubbles. By preparing extra assay and sample mix in the 96 well plate, air bubbles should be avoided using this pipetting method.
12. Following addition of all samples and assays, follow the detailed protocol for loading the Chip on the IFC mixer and then running the Chip on the Biomark HD. (See below). Note: The Camera requires time to turn on (~20 min) make sure to turn on early. Also, loading the Chip takes ~ 1 hr 20 min and running the chip takes ~1 hr and 30 min.

IV. Pipetting Map for 48.48

V. Pipetting Map for 96.96

1.2 Cloning protocols

1.2.1 Gibson assembly

Gibson assembly allows us to create a construct out of overlapping fragments.

Protocol

1. Design the assembly. Online tools such as [NEB Builder²](http://nebuilder.neb.com/) and built-in tools such as those in SnapGene are helpful for designing this.

Note: If designing Gibson primers manually, having an overlap region of at least 30bp is recommended. Each fragment must also be over 200bp.

2. Generate fragments by either PCR followed by DpnI digest, or restriction-digest of a target template.

Note: Skipping the DpnI digest step will often still work, but will tend to increase the number of background colonies because the source plasmid is still around. If the cloning strategy selects using a different antibiotic resistance cassette, it may be possible to skip this digest step.

3. Setup a reaction with 150 ng of the vector fragment, followed by a 2x molar ratio of each of the inserts. Add 2x by-volume Gibson master mix/HIFI assembly mix solution.
4. Incubate at 50°C for 15 minutes (2-3 fragments) or for 60 minutes (4-6 fragments).
5. *Transform competent cells* (page 14) with 2 uL of the Gibson mixture.
6. Store unused Gibson products at -20°C

Example

If you are cloning with pENTR and your vector concentration is 50 ng/ul, your insert is 1 kb and 50 ng/ul, then you want 3 uL of pENTR and 0.75 uL of insert. The Gibson mix (NEB HIFI Assembly Mix) is 2x. So you would add 3.75 uL of the assembly mix to the plasmid and insert mix. Following the assembly directions, place at 50°C for 15 minutes (1-2 inserts), followed by transformation in competent cells.

² <http://nebuilder.neb.com/>

1.2.2 LR cloning

Estimated time: 1 hr

Protocol

Note: In most instances, a 5 uL reaction is sufficient.

1. Add 50-150 ng of entry plasmid and 150 ng of destination plasmid. Dilute with water to 4 uL or 8 uL.
2. Thaw LR Clonase II enzyme mix on ice for two minutes. Vortex the enzyme mix briefly (2 seconds each time).
3. Add 1 uL clonase mix for a 5 uL reaction, or 2 uL clonase for a 10 uL reaction.
4. Incubate at 25 degrees C for 1 hour.
5. **Optional, if transforming immediately:** Add 0.5/1.0 uL Proteinase K to each sample to terminate the reaction. Incubate at 37 degrees C for 10 minutes.
6. Transform competent cells.

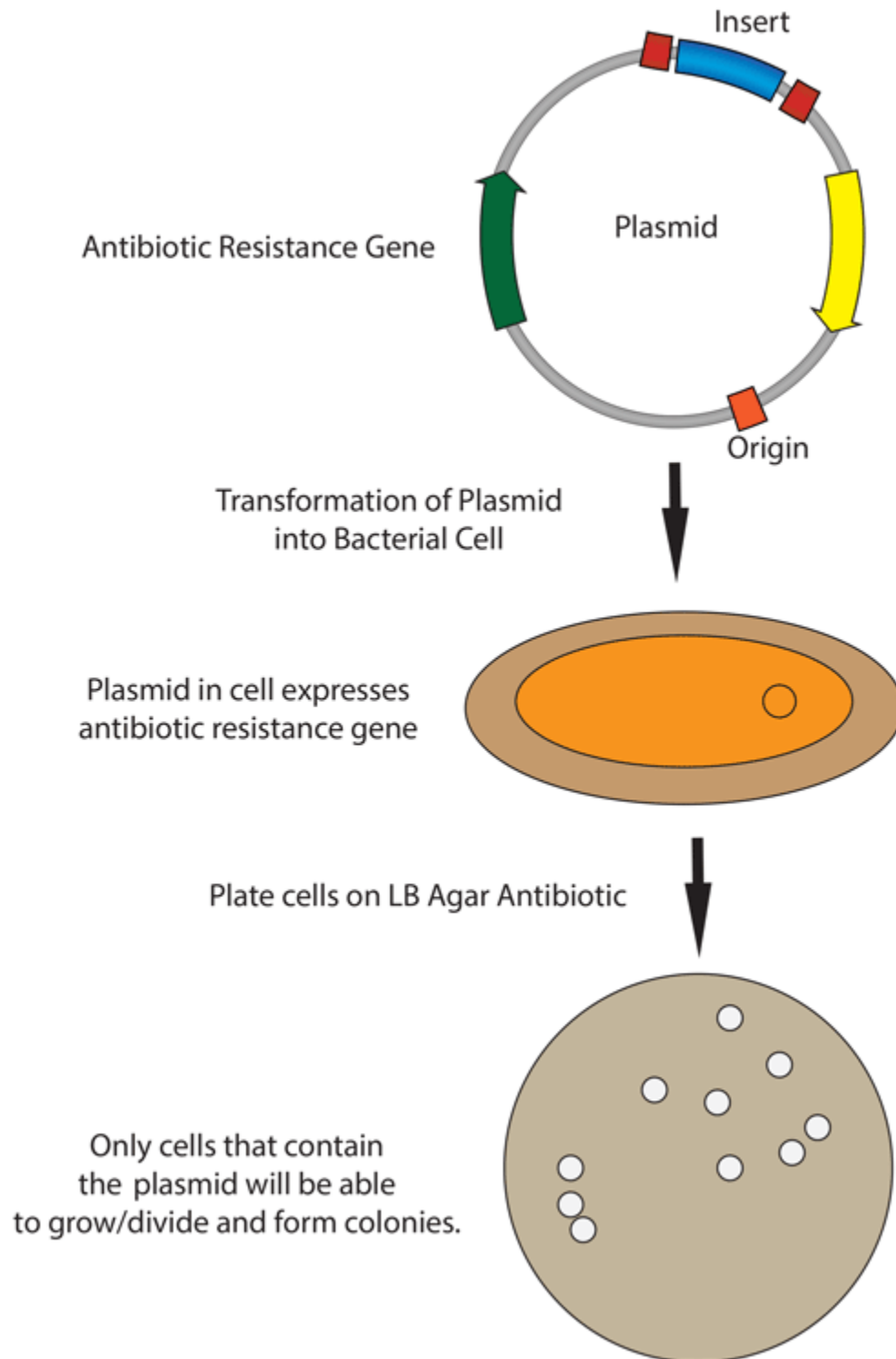
Expected results

An efficient reaction will produce > 5000 colonies, if the entire volume is transformed and plated.

1.2.3 Bacterial transformation

Cloning summary

In molecular cloning, we transform DNA of interest into competent bacterial cells. There are several different competence methods that require different transformation methods. After transformation, bacteria that have incorporated the DNA of interest are selected for using various antibiotics.



Heat-shock transformation (chemically competent bacteria)

Estimated time: 1 to 2 hours

1. Take chemically competent cells out of -80°C and thaw on ice (approximately 20-30 mins).
 2. Remove agar plates (containing the appropriate antibiotic) from storage at 4°C and let warm up to room temperature and then (optional) incubate in 37°C incubator.
 3. Mix 1 - 5 ul of DNA (usually 10 pg - 100 ng) into 20-50 uL of competent cells in a microcentrifuge or falcon tube. GENTLY mix by flicking the bottom of the tube with your finger a few times.
-

Note: Transformation efficiencies will be approximately 10-fold lower for ligation of inserts to vectors than for an intact control plasmid.

4. Incubate the competent cell/DNA mixture on ice for 20-30 mins.
 5. Heat shock each transformation tube by placing the bottom 1/2 to 2/3 of the tube into a 42°C water bath for 30-60 secs (45 secs is usually ideal, but this varies depending on the competent cells you are using).
 6. Put the tubes back on ice for 2 min.
 7. Add 250-1,000 ul LB or SOC media (without antibiotic) to the bacteria and grow in 37°C shaking incubator for 45 min.
-

Note: This outgrowth step allows the bacteria time to generate the antibiotic resistance proteins encoded on the plasmid backbone so that they will be able to grow once plated on the antibiotic containing agar plate. This step is not critical for Ampicillin resistance but is much more important for other antibiotic resistances.

8. Plate some or all of the transformation onto a 10 cm LB agar plate containing the appropriate antibiotic.
-

Note: If the culture volume is too big, gently collect the cells by centrifugation and resuspend in a smaller volume of LB so that there isn't too much liquid media on the agar plates. If the agar plate doesn't dry adequately before the cells begin dividing, the bacteria diffuse through the liquid and won't grow in colonies.

9. Incubate plates at 37°C overnight.
-

Note: Use 30 °C if you are working with unstable vectors (e.g. repeats and/or viral vectors)

1.3 Protein production

1.3.1 Protein gel casting

The recipes and procedures on this page were originally posted [here](#)³.

Required stock solutions

- **40% Acrylamide stock solution:** Solution of monomers for gel polymerization.

We find it cheaper to buy premixed 40% stock solution, with a acrylamide:bis-acrylamide ratio of 29:1 (3.3%). Stocks with a 37.5:1 ratio also work, and are typically used for resolving larger proteins.

- **3x bis-Tris gel buffer:** Ion buffer used in gel casting.

Component	Concentration	g/L to final concentration
bis-Tris	1 M	209.242
HCl	Add to pH 6.5-6.8	

- **10% APS:** One of the polymerization initiators. Only a small quantity needs to be prepared; each gel only requires 25 uL.

Component	Concentration	g/L to final concentration
Ammonium persulfate	10%	100

Casting protocol

Warning: The acrylamide monomers used here are toxic. Read the [SDS](#)⁴.

Perform polymerization steps with a lab coat in a fume hood, and collect rinse waste in a waste container.

1. Prepare 1X resolving and stacking buffers. These buffers can be stored in the refrigerator for several weeks. Recipes given here for enough for 2 0.75mm gels.

Resolving buffer: ~3 mL per gel (6.5 mL total):

Component	Volume	Final concentration
3x bis-Tris gel buffer	2.2 mL	1x
40% Acrylamide stock	1.3 mL	8%
DI water	3 mL	

³ https://openwetware.org/wiki/Sauer:bis-Tris_SDS-PAGE,_the_very_best

⁴ <https://www.fishersci.com/store/msds?partNumber=BP14081&productDescription=ACRYLAMIDE%3ABISACRYLAMIDE+29%3A1&vendorId=VN00033897&countryCode=US&language=en>

Stacking buffer: ~1.2 mL per gel (2.5 mL total):

Component	Volume	Final concentration
3x bis-Tris gel buffer	0.83 mL	1x
40% Acrylamide stock	0.32 mL	5%
DI water	1.36 mL	
Bromophenol blue		50 uL (enough to give color)

Gel casting setup

In-lab, we have the ability to cast two gels simultaneously; this is recommended even if you only need one, so that you have a backup in case of pouring mishaps. Our gel runner also requires two poured gels to properly seal.

1. Locate two 0.75mm spacer plates and two short glass plates.
2. Use ethanol and a Kimwipe to clean both glass surfaces.
3. Assemble them in the green alignment device.
4. Lock the two gels into the transparent gel pouring device.

Resolving gel

1. Measure 6.5 mL of **1x resolving buffer** per gel to pour.
2. Add 50 uL of **10% APS** per gel, mix well.
3. Add 20 uL **TEMED**, mixing quickly. Pour both gels to the resolving gel height (3 mL per gel). Lightly tap and tilt the gel to remove any bubbles.
4. Once done pouring, quickly but carefully fill the remaining height with water, making sure the gel-water interface stays undisturbed.
5. Wait for the polymerization reaction to finish (noticeable by a change in refractive index).
6. Drain the water by tilting the gel past 90 degrees, and wicking away with a Kimwipe.

Stacking gel

1. Measure 2.5 mL of **1x stacking buffer** to pour.
2. Add 20 uL of **10% APS**, mix well.
3. Add 10 uL **TEMED**, mixing quickly. Fill the top of the gels until just before overflowing. Insert the comb into the top, letting it rest on the spacers.
4. Wait for the stacking gel to polymerize.
5. Rinse with water to remove unpolymerized acrylamide.

6. If removing the combs prior to storage, slowly remove the comb, ensuring that wells are not broken.

1.3.2 Chitin-binding protein purification

Required solutions

- Chitin Lysis buffer:

Component	Concentration	amount/L final volume
DI water	89.8%	898 mL
HEPES	20mM	4.766 g
NaCl	800 mM	46.762 g
EDTA	1 mM	0.292 g
Triton-X100	0.2%	2 mL
Glycerol	10%	100 mL
KOH	to pH 7.2	

- Chitin cleavage buffer:

Component	Concentration	amount/L final volume
Chitin lysis buffer	Base	898 mL
DTT	100 mM	15.423 g

- Chitin stripping buffer:

Component	Concentration	amount/L final volume
Chitin lysis buffer	Base	898 mL
SDS	1%	10 g

- 2x Dialysis buffer:

Component	Concentration	amount/L final volume
DI water	79.8%	798 mL
HEPES	100mM	23.83 g
NaCl	200 mM	11.69 g
EDTA	0.2 mM	0.0584 g
DTT	2 mM	0.3085 g
Triton-X100	0.2%	2 mL
Glycerol	20%	200 mL
KOH	to pH 7.2	

- 10% PEI: pH to 7.2

Protocol

1. Resuspend the cell pellet in lysis buffer plus cOmplete protease inhibitor, on ice. For every gram of cell pellet, use 10 mL lysis buffer.
2. Sonicate the resuspended cells, using 5 cycles of 30-second, medium-amplitude 50% duty-cycle sonication.
3. Spin to clarify the lysate using maximum centrifuge speed (14600 xg) for 30 minutes.
4. If the protein of interest has DNA-binding activity, add 1.2 mL/10mL **10% PEI** dropwise while stirring. Centrifuge at maximum speed for 30 minutes to remove bound DNA.
5. Incubate the supernatant with 10 mL chitin resin / 500 mL original culture. Tube rotate for 1 hour at 4C.
6. Wash with 10-20 CVs of lysis buffer.
7. Add cleavage buffer. Incubate at 4C for 48 hours.
8. Elute into separate 1 mL fractions.
9. Dialyze elution buffer into 2x dialysis buffer, followed by concentration.

Regeneration Protocol

1. Wash the resin with 3CVs of stripping buffer.
2. Stop the flow, and soak the resin in stripping solution for 30 minutes.
3. Wash with an additional 10 CVs of stripping buffer.
4. Rinse with 20 CVs of water.
5. Equilibrate with 5 CVs of 20% ethanol.
6. Store the resin in 20% ethanol at 4C.

1.3.3 Denaturing protein gel run and staining

Solutions required

- **20x MES-SDS running buffer stock solution:** Suitable for separating proteins with a molecular weight less than 75 kDa.

It is also generally cheaper to order this as a pre-mixed 20x stock solution. If you need to make it yourself, the recipe is:

Component	Final concentration	g/L to final concentration
MES	1 M	195.2 g
Tris	1 M	121.13 g
EDTA	20 mM	5.845 g
SDS	2%	N/A

- **200x running buffer reductant:** Ensures that the gel remains under reducing conditions when run. Add directly to 1x running buffer before filling the gel tank.

Component	Final concentration	g/L to final concentration
Sodium bisulfite	1 M	104.061 g

- **200 mM Tris-HCl stock:**

Component	Concentration	g/L to final concentration
Tris-HCl	200 mM	31.52 g
NaOH	Add to pH 6.8	

- **10% SDS stock:** At low temperatures, the SDS may fall out of solution, but will redissolve when mixed with the other ingredients. Mix well before transferring.
- **0.1% bromophenol blue:** 1 mg / mL
- **2x sample buffer:** Used to denature and solubilize protein samples. Can be stored

Component	Final concentration	Volume / 10 mL
200 mM Tris-HCl stock	100 mM	5 mL
Glycerol	20%	2 mL
10% SDS stock	2%	2 mL
0.1% bromophenol blue stock	0.01%	1 mL

- **Coomassie staining dye:** When preparing this dye, pour the 10% methanol first, using it to dissolve the R-250. Then, add water. Add the glacial acetic acid last to prevent aggregation.

Component	Final concentration
Coomassie R-250	0.2% (2g/L)
Methanol	10%
Acetic acid	10%
Water	80%

Running procedure

1. Dilute enough **20x MES-SDS running buffer** to fill the gel tank, adding fresh **200x running buffer reductant** if a gel has not been recently run.
2. Add **2x sample buffer** to the protein sample of interest. Add 2-mercaptoethanol to a final concentration of 1%.

Warning: 2-mercaptoethanol smells awful; always add it inside a fume hood.

3. Briefly heat the protein sample to above 80C for one minute, to ensure full protein denaturation.
4. Place a *prepared bis-Tris protein gel* (page 17) in the gel-runner, and fill both chambers with the prepared 1% MES-SDS running buffer.
5. Carefully load samples, including a protein standard.
6. Run the gels at constant current, about 30 mA per mini-gel. The dye band runs around 3-5 kDa, so it is typically ok to run the dye band to the bottom of the gel unless very small proteins are of interest.

Staining

1.3.4 Protein expression

Estimated time: 2 days

Protocol

Note: The BL21 strain of *E. coli* is best for protein production. It has several proteases knocked out, among other modifications. There are several sub-strains that can be relevant:

- **BL21:** The classic, useful if your promoter is a standard *E. coli* promoter such as *lac*, *tac*, *trc*, *ParaBAD*, *PrhaBAD*, or *T5*.
- **BL21(DE3)** : A strain with integrated T7 polymerase. Can be used in place of **BL21** but is required if using the *T7* or *T7-lac* promoters.

- **BL21-RIL** : A strain with extra copies of certain tRNAs, in order to correct for the rarity of some codons in *E. coli*. This can be helpful in upping protein titers, especially when expressing some mammalian proteins.
-

1. In the morning, start a **5 mL** culture of your strain, in LB/antibiotic media.
 2. Prepare *TB media* (page 42) in Erlenmeyer flasks. For proper aeration, flasks should be filled to at most 30% the listed volume.
 3. Before leaving for the day, inoculate **20 mL** LB/antibiotic cultures with **400 uL** of the 5 mL culture.
 4. The following morning, use the NanoDrop to calculate the OD of the overnight cultures. Inoculate large flasks to an OD600 of 0.075 (e.g. three doublings to get to 0.6).
-

Note: Remember to take a sample of LB/TB media to properly blank the NanoDrop!

Induction to OD600 0.075 means that each 20 mL culture can induce around 1L of TB media, as the overnight OD600 should be around 5-10. If you are inducing more than 1L, you may need to make multiple 20 mL cultures.

Calculate OD's with the simple dilution formula:

$$\text{mL starter culture} = \frac{\text{mL bulk culture} \cdot 0.075}{\text{OD of starter culture}}$$

5. Grow cultures until an OD600 of 0.6. Sample every half hour to an hour. It should take around 90 minutes to 2 hours to reach an OD of 0.6.
6. Lower the temperature to 30C and induce with 0.5 mM sterile-filtered IPTG.
7. Let the culture grow for 5-6 hours.
8. Harvest cells via centrifugation at 4000xg for 30 minutes. At this point, the cell pellet is immediately ready for further protein purification steps. The cell pellet can also be stored at -20C or -80C.

1.3.5 His-tag protein purification

Required solutions

- Lysis buffer:
- 10% PEI: pH to 7.2

Note: Do not use the cOmplete running and elution buffers with normal Ni-NTA His-tag resin! They contain chelating agents that will remove the nickel ions! cOmplete His-tag resin is resistant to small concentrations of chelating agents.

- cOmplete running buffer (native):

Component	Concentration	g/L final volume
DI water	90%	
HEPES	20mM	4.766 g
NaCl	800 mM	46.762 g
Imidazole	20 mM	1.362 g
EDTA	1 mM	0.292 g
DTT	2 mM	0.3085 g
Glycerol	10%	
NaOH	to pH 7.2	

Note: Perform the pH after all components have been added.

- cOmplete elution buffer (native):

Component	Concentration	g/L final volume	Purpose
DI water	90%		
HEPES	20mM	4.766 g	Main buffer component
NaCl	800 mM	46.762 g	High salt maintains protein solubility
Imidazole	300 mM	20.42 g	Prevents non-specific binding to column.
EDTA	1 mM	0.292 g	Chelating agent, deactivates proteases and other enzymes.
DTT	2 mM	0.3085 g	Reducing agent, prevents nonspecific cystine crosslinking.
Glycerol	10%		Prevents hydrophobic protein-protein interactions.
NaOH	to pH 7.2		

Note: Perform the pH after all components have been added.

- **NI-NTA running buffer (native):**

Component	Concentration	g/L final volume	Purpose
DI water	90%		
NaH ₂ PO ₄	50mM	6.90 g	Main buffer component.
NaCl	800 mM	46.762 g	High salt maintains protein solubility.
Imidazole	15 mM	1.022 g	Prevents non-specific binding to column.
beta-mercaptoethanol	5 mM		Weaker reducing agent, prevents nonspecific cystine crosslinking.
Glycerol	10%		Prevents hydrophobic protein-protein interactions.
NaOH	to pH 8.0		

Note: Perform the pH after all components have been added.

Protocol

Protocol

1. Resuspend the cell pellet in lysis buffer, on ice. For every gram of cell pellet, use 10 mL lysis buffer.
2. Sonicate the resuspended cells, using 5 cycles of 30-second, medium-amplitude 50% duty-cycle sonication.
3. Spin to clarify the lysate using maximum centrifuge speed (14600 xg) for 30 minutes.
4. If the protein of interest has DNA-binding activity, add 1.2 mL/10mL **10% PEI** dropwise while stirring. Centrifuge at maximum speed for 30 minutes to remove bound DNA.
5. Fill a column with roughly 1.0 mL resin per gram of cell lysate.

Note: When using affinity columns, the volumes required will be listed as Column Volumes (CVs). The amount of resin is the CV. For a gram preparation, this would mean the CV is 1 mL.

5. Equilibrate the column with 3 CVs of running buffer.
6. Flow the clarified cell lysate across the column, collecting the flow-through.
7. Rinse the column with 8 CVs of running buffer, collecting flow-through fractions.
8. Elute with four separate 0.5 CV washes of elution buffer.

9. Run all collected samples on a SDS-page gel to confirm successful purification.

1.4 TC protocols

1.4.1 Adherent Cell Culture

General culture volumes⁵

Culture dish/plate/flask	Approx. volume media
10-cm	10 mL
15-cm	15 mL
6-well	2 mL (1-3 mL)
12-well	1 mL (1-2 mL)
24-well	500 μ L (500-1,000 μ L)
96-well	100 μ L (100-200 μ L)
T-75	10 mL (8-15 mL)
T-182	25 mL (>25 mL)

General culture handling procedure

- Lift lid and tilt plate with one hand while aspirating/pipetting with the other.
- The lid should cover the plate as much as possible, providing just enough opening to perform the task
- The lid opening should face away from the hood opening (i.e., away from you and toward the back wall).
- Tilt the plate toward the back when aspirating.
- When aspirating, immerse the pipet tip in the media but don't touch the plate bottom until the very end to avoid aspirating away cells.
- Tilt media gently against the side wall of the dish to avoid dislodging cells.

Changing media

Unless you have very specific experimental conditions, it is good practice to check on your cultures every 2 or 3 days and change media if the cells are not ready to be split or expanded.

⁵ <https://www.thermofisher.com/us/en/home/references/gibco-cell-culture-basics/cell-culture-protocols/cell-culture-useful-numbers.html>

Passaging cells

Passage number is an “indicator” of cells age. Every time the cells are detached from one dish and passaged into another dish, the passage number increases by 1.

1. Aspirate away old media.
2. Wash with PBS (*optional*)
 - i. Gently add PBS to the side of the dish/plate, not directly on the cells
 - ii. Aspirate away PBS

Note: Some types of cells do not require trypsin to detach (e.g. HEK293Ts, Plat-Es). In that case, just pipette up and down until cells detach. This looks like a thin film disappearing from the bottom of the dish/plate/flask.

3. Add 0.05% trypsin (dilute in PBS, DO NOT DILUTE WITH ANYTHING WITH FBS!!!) to detach cells from the plate
 - Trypsin should cover the entire surface area, general volumes are as follows:

Culture dish/plate/flask	Approx. volume 0.05% Trypsin
10-cm	1 mL
15-cm	3 mL
6-well	500 μ L/well
12-well	300 μ L/well
24-well	150 μ L/well
96-well	why are you trying to trypsinize a 96-well o_o
T-75	5 mL
T-182	15 mL

- i. Incubate in 37°C incubator for 3-4 min, keep checking cells to avoid over-digestion.
 - Detached cells look rounded and refractile under the microscope.
 - Different types of cells will require different incubation periods.
 - The stronger the trypsin (higher %) the faster cells will detach.
 - ii. Neutralize with >2 volumes of media that contains FBS (e.g. neutralize a 1 mL 0.05% trypsin with >2 mL FBS-containing media).
 - iii. Mix well by pipetting and make sure that all cells are detached.
4. If needed, take a known volume of cells and count cells using hemocytometer.
5. Seed new dish/flask with desired number or dilution of cells (make sure cells in the original dish are well resuspended before you take an aliquot to seed the new dish).
6. Cultures should be labeled with cell line name, date, passage number and initials of the owner.

- Certain cell lines also need passage # of antibiotic selection, e.g. “Plat-E, Blast/Puro @ P12”

1.4.2 Adherent Cell Staining

Protocol

1. Remove media and add cold 4% paraformaldehyde (PFA) (*in fume hood*)
2. Incubate cells in 4% PFA for 1 hour at 4 degrees C.
3. Remove 4% PFA (*in fume hood*)
4. Wash cells 3 times with cold PBS. (*cells may be left in PBS overnight at 4 degrees C*)
5. **If staining nuclear proteins**, change solution and incubate cells in 0.5% PBS/Tween for **1 hour to overnight** at 4 degrees C.

Note:

1. Never remove all of the media from the cells-they will dry and absorb antibody in a non-specific way, thereby skewing your results.
 2. Do not breathe in the PFA; it is toxic!
 3. If mounting, try to avoid bubbles in the final step. Leave some solution still on your slides so the cells do not dry and carefully dispense 2-3 drops of the solution, very carefully putting the coverslip on your slide.
 4. Unless otherwise noted, everything is performed at room temperature.
 5. Keep your antibodies on ice at all times.
 6. Be very gentle with neuronal cultures, they are not very adherent.
-

1.4.3 Cryopreservation of mammalian cells

Cryopreservation Medium (should always be made fresh)

Growth media containing FBS (FBS should be added for freezing if growth media is serum-free) Final concentration of FBS should not exceed 20%. FBS binds toxic materials released if some cells are lysed during the freezing or thawing process. A cryopreservative MUST also be used: Sterile dimethyl sulfoxide (DMSO) at a final concentration of between 7% and 10% (preferably 10%). OR Sterile glycerol at a final concentration of 10%. Glycerol should be used if the cell type to be cryopreserved may be adversely or unwantedly affected by exposure to strong bi-polar compounds such as DMSO. Important: The cryopreservatives, especially DMSO, must be used fresh each time. If you buy large volumes, transfer a small amount to another tube, wrap in aluminum foil (DMSO is light sensitive) and use the aliquot. Minimize opening and closing the reagent bottle. This is to avoid oxidization and/or absorption potentially toxic materials from the air.

1.4.4 Important Tissue Culture Practices

- Wash your hands before and after working with cell cultures.
- Clean pipettes, pipette aids, markers, tube racks, and the work surface immediately before and after use using 70% EtOH.
- When handling mammalian cells always work in the Biosafety cabinet.
- Before starting your cell culture work, open the Biosafety cabinets and let the laminar air flow run for 5-10 minutes.
- Spray the cabinet with 70% ethanol and let sit for 1-2 mins then wipe your work surface using kimwipes (not paper towels).
- Always wear a lab coat and gloves when working in the BS cabinet.
- Only have items you need in the BS cabinet- Having too many items in the BS cabinet interrupts the laminar air flow and can cause contaminants to be introduced into the BS cabinet.
- Do not place items on the BS cabinet grill- this created a gap in the air laminar flow.
- Always keep reagent bottles, tubes and culture dishes/flasks closed when not in use.
- Any spillage inside and outside of the hood should be immediately wiped up and cleaned.
- Be sure you are working INSIDE the BS cabinet- not on or above the grill.
- Do not work above/over open reagents/ bottle caps... etc. Open containers should be put far away from you such that no work is done above/over them.
- Do not use your phone or computer while handling cells- this leads to contaminations!
- All TC room equipment (e.g., tube racks, timers, EtOH bottles, gloves) should stay in the TC room. Also, outside equipment should not be brought into the TC room unless absolutely necessary.
- Do not discard your culture with the media in them, aspirate media before putting them in the Biohazard waste.
- In order to minimize contamination and cross contamination, ethanol should be run through the aspirator tip after use.
- Always check the media and cells before using the BS cabinet - turbid media usually means that the culture is contaminated.
- If your culture is contaminated:
 - i. Discard it immediately in a biohazard waste bin and close and change the bin. Change your gloves immediately. Decontaminate any surfaces the contaminated culture came into contact with using bleach or ethanol.
 - ii. Clean any surfaces in the incubator where the contaminated culture was stored (Do not directly spray ethanol in the incubator, instead wet a Kimwipe with ethanol and clean the surface).

- iii. Check the media bottle you used for the contaminated culture - if suspicious, discard the media. Change gloves again and continue working with the rest of your cultures.

1.4.5 Tissue Culture Media

Fetal Bovine Serum (FBS) Aliquots

- FBS contains many nutrients and growth factors to support cell growth. To preserve FBS, it is usually stored in the -20C freezer until used. To avoid multiple freeze-thaw cycles of FBS and to prevent contaminations we make FBS aliquots.
- Thaw 500mL FBS bottle in the fridge overnight
- Make 10 aliquots of 50mL each. Be sure to stop at the 50mL mark. Having larger volumes may cause the FBS to overflow (and be contaminated) when frozen again (FBS volume expands when frozen).
- Store aliquots in -20C freezer in the TC room.
- Take responsibility for aliquoting new stocks, when older stocks are almost used up. Make sure to use sterile technique in preparing aliquots.

Pen-strep aliquots

- Penicillin-streptomycin is a mixture of penicillin and streptomycin and is added to the media to prevent bacterial contamination. However, it is always important to be very careful when handling cultures as contamination may still occur even when pen-strep is added.
- Thaw 50mL pen-strep bottle in the fridge overnight
- Pen-strep is aliquoted into 5mL aliquots and stored in the TC -20C until used.
- Take responsibility for aliquoting new stocks, when older stocks are almost used up. Make sure to use sterile technique in preparing aliquots.

Preparation of basic 2D cell culture media (DMEM + 10% FBS +1% Pen-strep)

- Thaw 50mL of FBS and 5mL of penicillin-streptomycin aliquots.
- To a 500mL DMEM bottle, add 50mL of FBS and 5mL of penicillin-streptomycin.
- Mix solution and store in 4C in the TC room.

Note: this media mixture is commonly used but it is not the only type of media used for 2D cell culture. You should find out the recommended type of media used for the culture of the cell type you are using.

1.4.6 Plat-E Retroviral Transduction of MEFs

Plat-Es are a retrovirus packaging line that produce retroviruses that can only infect mouse cells. Plat-E cells contain gag, pol and env genes, allowing retroviral packaging with a single plasmid transfection. Supposedly, Plat-E viruses can't be frozen.

Plat-E Transfection

Day One (seed Plat-Es):

1. Seed Plat-E cells at 1 million cells/well onto 6 well plates after 0.1% gelatin coating (10 min).

Tip: If starting from frozen, **start growing Plat-E cells 1 week prior** - they will be slow growing at first (don't change culture medium during the first 3 days). Split Plat-Es 4X-6X every 2-3 days when culture reaches 70-90% confluency.

Day Two (transfect Plat-Es):

1. Make a mastermix (MM) of PEI and Knockout DMEM as follows (MM for 1 well/each virus):
 - a. For *just 1 well of a 6-well plate*:
 - i. In a 1.5 mL tube, **FIRST** add 180 μ L KO DMEM.
 - ii. **SECOND**, add 9.00 μ L PEI (1 mg/ml) for a 5:1 ratio (of μ g PEI: μ g DNA ratio).
 - iii. Gently flick to mix. Let sit for 5 minutes.
 - b. For *3.5 rxns or wells of a 6-well plate*, same as above but use 630 μ L KO DMEM + 31.5 μ L PEI (1 mg/ml) for a 5:1 ratio

Important: Ensure that you **add PEI to KO DMEM, NOT the other way around!!** Also make sure to use KO DMEM as KO DMEM facilitates lipid-mediated transfections (e.g. PEI) which might be disrupted by regular DMEM.

Note: PEI optimal concentration varies by batch and must be tested before transfection. Current batch is 5:1 as of 2020.02.19 (typical range 4:1 to 6:1).

2. Add DNA to this MM as follows, then mix and wait 15 minutes:
 - a. 1.8 μ g DNA/well, (or 6.3 μ g DNA total for 3.5 rxns)

Component	Volume/ 1 rxn	Volume / 3.5 rxn	Total Vol/well*
Brn2 (4.49 $\mu\text{g}/\mu\text{L}$)	0.40 μL	1.40 μL	189.40 $\mu\text{L}/\text{well}$
Ascl1 (2.78 $\mu\text{g}/\mu\text{L}$)	0.65 μL	2.27 μL	189.65 $\mu\text{L}/\text{well}$
Mytl-1 (2.1 $\mu\text{g}/\mu\text{L}$)	0.86 μL	3.00 μL	189.86 $\mu\text{L}/\text{well}$
Ngn2 (4.4 $\mu\text{g}/\mu\text{L}$)	0.41 μL	1.43 μL	189.41 $\mu\text{L}/\text{well}$
mIsl-1 (6.2 $\mu\text{g}/\mu\text{L}$)	0.29 μL	1.02 μL	189.29 $\mu\text{L}/\text{well}$
Lhx3 (5.4 $\mu\text{g}/\mu\text{L}$)	0.33 μL	1.17 μL	189.33 $\mu\text{L}/\text{well}$
p53DD (3.93 $\mu\text{g}/\mu\text{L}$)	0.46 μL	1.60 μL	189.46 $\mu\text{L}/\text{well}$
hRasG12V (0.9 $\mu\text{g}/\mu\text{L}$)	2.00 μL	7.00 μL	191.00 $\mu\text{L}/\text{well}$
NIL (7.35 $\mu\text{g}/\mu\text{L}$)	0.24 μL	0.86 μL	189.24 $\mu\text{L}/\text{well}$

3. Add each KO DMEM + PEI + DNA mix to a single 6-well (1 WELL PER VIRUS) **DROPWISE** and evenly around the plate, rocking the plate back and forth, side to side to mix. Place back in incubator. *Rocking side to side prevents cells from clustering at the well edge.*

Day Three (Plat-E media change + seed MEFs):

1. Change with 1.25 mLs fresh media (DMEM + 10% FBS) after 24 hours.
2. Seed MEFs
 - i. Coat wells in 0.1% gelatin for approx. ~10 min.
 - ii. Seed at ~5-10K cells per well of 96-well plate. For larger sizes, scale accordingly (e.g. 96-well size is about 1/8 of 12-well, seed ~60K for 12-well).

Day Four (Plat-E media change + infect 1):

1. Harvest media after another 24 hours and PROCEED TO TRANSDUCTION!

Transduction of Mouse Embryonic Fibroblasts (MEFs)

Day Four (Plat-E media change + infect 1):

1. Transduce MEFs with retroviruses made from the Plat-E cells

Note: Each virus will make ~1.25 mL/well from each 6-well of Plat-E (enough for 1 96-well plate). 11.375 μL of each virus will be added to each well of a 96-well plate **ALONG WITH POLYBRENE** (1,000X at 5 mg/mL)

- a. Filter each virus through a filter
 - i. For 6F, all 6 TFs (BAMNIL) will be filtered through the same syringe together to simplify adding them each individually to MM combos.
- b. Master mixes will be made for simpler “aliquoting” into wells. The following table is a guide for the final total volume for each well depending on the plate.

Culture plate	Total DMEM (i.e. MEF media) + virus per well
6-wells	2 mL
12-wells	1 mL
24-wells	500 μ L
48-wells	250 μ L
96-wells	100 μ L

Note: You can either 1.) filter each virus then mix together (minimizes filtering) or 2.) mix altogether then filter (standardizes mixing). Because filtering is the most annoying step, it is advised to minimize filtering.

2. Examples of mixing AFTER filtering

i. Example - 6F alone (96-well = 100 μ L total/96-well):

For 1 rxn, 96-well: 68.25 μ L 6F (= 11.375 μ L PER FACTOR*6) + 31.75 μ L DMEM + 0.1 μ L polybrene (1,000X) = 100 μ L total/96-well

For 3.5 rxn, 96-well: 238.9 μ L 6F + 111.1 μ L DMEM + 0.35 μ L polybrene (1,000X) = 350 μ L total for 3.5 96-wells

ii. Example - 6F + DD + RR (96-well = 100 μ L total/96-well):

For 1 rxn, 96-well: 68.25 μ L 6F + 11.375 μ L p53DD + 11.375 μ L hRasG12V + 9 μ L DMEM + 0.1 μ L polybrene (1,000X) = 100 μ L total/96-well

For 3.5 rxn, 96-well: 238.9 μ L 6F + 39.8 μ L p53DD + 39.8 μ L hRasG12V + 31.5 μ L DMEM + 0.35 μ L polybrene (1,000X) = 350 μ L total for 3.5 96-wells

3. Add virus mixes to each well dropwise, rocking back and forth to mix.

4. Add 1.25 mL fresh media (DMEM + 10% FBS) to Plat-E plates for a second time.

Day Five (infect 2):

1. Collect media from Plat-Es again and reinfect/retransduce the plates for a second day.

Day Six (1 dpi):

1. Change media on transduced MEFs according to transduction MM table (e.g. 100 μ L for 96-well)

Day Seven (2 dpi):

1. Add glia cells to the transduced plates of fibroblasts.

a. 1 vial of glia = 3 flasks, able to use approximately one flask per FULL PLATE of 96 wells.

b. 500 μ L will be added to the 24-well plates.

Day Eight (3 dpi):

1. Media change plates to N3 media

- a. N3 media = N3 base + BDNF/CNTF/GDNF (1,000X, 10 $\mu\text{g/mL}$) + FGF10 (10,000X, 100 $\mu\text{g/mL}$) + 2% FBS (*optional*)

2. **Spike in 1,000X RepSox to N3 media for RR conditions**

Day 10, 12, 14, etc:

1. Change N3 media until cells are fixed with PFA for staining and imaging.

1.4.7 Transient 293T Transfection

Protocol

1. Seed confluent 293T cells approximately 1-2 days prior to transfection. Cells must be ~90% confluent for efficient transfection.
2. Make a mastermix of PEI and Knockout DMEM as follows:
 - a. For *one 10cm plate*, add 53.34µl PEI (1mg/ml; 4:1 ratio) TO 1.33mls Knockout DMEM in a 50ml conical, mix, let sit for 5 minutes.

Important: *Ensure that you add PEI to KO DMEM, not the other way around!*

3. Add DNA to this MM as follows, then mix and wait 15 minutes:
 - a. For *one 10cm retrovirus plate*, use 6.67ug viral plasmid + 6ug pIK-MLVgp + 0.667ug pHDMG.
 - b. For *one 10cm lentivirus plate*, use 6.67ug viral plasmid + 6ug pPAX2 + 0.667ug VSVG.

Tip: It's more efficient to make a master mix of the structural and packaging vectors (e.g. PAX2/VSVG or IK-MLV/HDMG) and aliquot that out.

4. Add this to each 10cm plate *dropwise* and evenly around the plate, rocking the plate back and forth, side to side to mix.
5. *After 24 hours*, change with 6.5mls fresh media (DMEM + 10% FBS).
6. Harvest media after another 24 hours and replace with another 6.5mls fresh media. Store media from each virus/vector in 50ml conical at 4°C (combine media from both plates into same tube!).
7. Harvest again after another 24 hours. May add the media from the two plates into the 50ml conical already being stored at 4C. After this second harvest, UV plates for about 30 minutes, then discard into biohazard.
8. Keep virus in 4°C until *ready to concentrate* (page 40)!

Note: Retroviruses are stable at 4°C for 2-3 days, with lentiviruses being stable for 4-5 days.

1.4.8 Virus concentration

Estimated time: 1 hour in-TC time, 1 day overnight time

Protocol

1. Filter virus through a 0.45um syringe filter into a 50ml conical. This eliminates any 293T cells that may have been carried over/collected in media.
2. Add 1/3 volume of Lenti-X concentrator (if 30ml virus, add 10ml Lenti-X). Will be approximately 8.7-9ml. Mix by inverting several times.
3. Store at 4°C for 24 hours (or overnight).
4. Centrifuge at 1500 x g at 4°C for 45 minutes, then remove the supernatant, careful to leave ~200-300ul media. For dsRed, remove almost all media, leaving only a minute amount to resuspend the pellet.
5. Resuspend gently in media using P1000.
6. Record volume of virus suspension.
7. Freeze in cryovial at -80°C.

RECIPES

2.1 Bacterial recipes

2.1.1 Antibiotic Stock Recipes

1000X Ampicillin (Amp)

1. Dissolve 100 mg of Ampicillin powder in 1 mL sterile/ELGA water (Or 1 g of Ampicillin powder in 10 mL sterile/ELGA water)
2. Sterile filter using a syringe filter
3. Make 1 mL aliquots and store in -20C

1000X Kanamycin (Kan)

1. Dissolve 50 mg of Kanamycin powder in 1 mL sterile/ELGA water (Or 500 mg of Kanamycin powder in 10 mL sterile/ELGA water)
2. Sterile filter using a syringe filter
3. Make 1 mL aliquots and store in -20C

1000X Chloramphenicol (Chlor)

1. Dissolve 34 mg of Chloramphenicol powder in 1 mL 70% EtOH (Or 340 mg of Chloramphenicol powder in 10 mL 70% EtOH)
2. No filtration step needed.
3. Make 1 mL aliquots and store in -20C.

2.1.2 Bacterial media recipes

TB (Terrific Broth)

The following recipe can be directly autoclaved in Erlenmeyer flasks if desired.

1. Combine and autoclave the following components. The final **10%** volume will be added afterwards.

Ingredient	Amount per 1L final volume
Tryptone	12 g
Yeast extract	24 g
Glycerol	4 mL
Deionized water	900 mL

2. After the mixture has cooled, add 100 mL / liter final volume of *10x TB salts* (page 42) and desired antibiotics.

10x TB salts

The potassium phosphate salt solution acts as a pH buffer. It must be autoclaved separately to prevent high-temperature reactions between the salts and other components of TB.

1. Combine the following, filling to the desired final volume with deionized water.

Ingredient	Target concentration	Amount per 1L final volume
KH ₂ PO ₄	0.17 M	23.1 g
K ₂ HPO ₄	0.72 M	125.4 g

Note: Because we are making a fairly concentrated solution, you will need to add less than the total final volume of water (e.g. for a liter of TB salts, you may only have to add 900 mL water).

I tend to add 70% of the water, wait until all of the salt dissolves, then fill up to the relevant line on the flask. This doesn't have to be super precise.

2. Autoclave the salt solution.

2.2 TC recipes

2.2.1 TC media recipes

N3 media

Used for neuronal differentiation during direct conversion.

- Total volume: **500 mL**

Component	%	Volume
Glutamax (1%)	1%	5 mL
N2 (100x stock)	100x stock	5 mL
B27 (50x stock)	50x stock	10 mL
Pen/Strep (1%)	1%	5 mL
FBS (<i>optional</i>)	2%	10 mL
DMEM F12		add to 500 mL

Add the following small molecules to small volumes immediately before media addition, to a **final volume of 10 ng/mL**:

- GDNF, from 1,000x stock (10,000 ng/ml)
- BDNF, from 1,000x stock (10,000 ng/ml)
- CNTF, from 1,000x stock (10,000 ng/ml)
- FGF10, from 10,000x stock (100,000 ng/ml)
- RepSox, **if in human cells**, from 1000x stock.

TODO

Check N2/B27 interpretation, and the RepSox stock concentration.

MEF media

Used for culturing MEFs and other fibroblasts.

Total volume: **500 mL**

Component	%	Volume
FBS	10%	50 mL
Pen/Strep (<i>optional</i>)	1%	5 mL
DMEM		add to 500mL

Breast cancer 1 media

Use for culturing the **LM2**, **BT-20**, and **MDA-MB-468** cell lines.

Component	Concentration	Volume / 50 mL
DMEM + 10% FBS	Base	Use MEF media
Anti-Anti	1%	500 uL

Breast cancer 2 media

Use for culturing the **HS 578T** and **BT-549** cell lines.

Component	Concentration	Volume / 50 mL
DMEM + 10% FBS	Base	Use MEF media
PSG	1%	500 uL
Insulin	10 ug/mL	125 uL of 4 mg/mL stock

Breast cancer 3 media

Use for culturing the **MDA-MD-231** cell line.

Component	Concentration	Volume / 50 mL
DMEM + 10% FBS	Base	Use MEF media
PSG	1%	500 uL

Breast cancer 4 media

Use for culturing the **SUM159** cell line.

Component	Concentration	Volume / 50 mL
F12	Base	47 mL
FBS	5%	2.5 mL
Anti-anti	1%	500 uL
Insulin	5 ug/mL	62.5 uL of 4 mg/mL stock
Hydrocortisone	1 ug/mL	
EGF	20 ng/mL	1 uL of 1 mg/mL stock

Glia media

Used for culturing glia cells

Total volume: **500 mL**

Component	%	Volume
Horse serum	10%	50 mL
Glucose	20%	100 mL
MEM	70%	350 mL

Sorting media

Used for preparing cells for sorting

Total volume: **500 mL**

Component	%	Volume
Pen/Strep	1%	5 mL
DMEM-F12	99%	495 mL

Collection media

Used for collecting cells during sorting.

Total volume: **500 mL**

Component	%	Volume
FBS	10%	50 mL
Pen/Strep	1%	5 mL
DMEM-F12	89%	445 mL

Dissociation media

Used for dissociating primary motor neurons harvested from spinal cords for plating/sorting.

Total volume: **6 mL**

Component	Volume
Papain	1 vial
DNase	1 vial
DMEM-F12	6 mL

TODO

Fill in details of vial size, verify amount of DMEM

Freezing media

TODO

Add freezing media recipe (missing in word doc)

TECH DOCUMENTATION

3.1 Repository setup

This repository is automatically built and pushed to <https://gallowaylabmit.github.io/protocols>

See the *Repository setup guide* (page 47) if you are interested in how the continuous build system works, or you'd like to fork this repository to make it your own.

TODO

Add info about labbot, repo setup, git, a more detailed RST primer, etc.

BOOTCAMP

4.1 IAP Bootcamp

4.1.1 Tissue culture basics

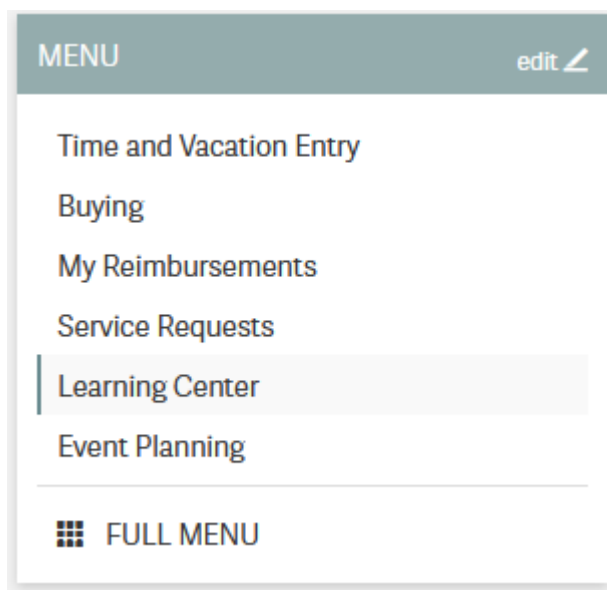
4.1.2 Day 0: Software and training setup

Our lab uses a lot of different software and various webservices. Before beginning IAP, you should install at least everything required on the list here, as well as completing all of the required EHS trainings.

EHS setup and trainings

Adding yourself to the lab's training group will register you with the EHS system and add all necessary trainings to your profile.

1. Go to <https://atlas.mit.edu> and go to the learning center, through the tab on the left:



2. In the upper right, select "My profile", then "Update PI/activities"

3. Add Kate E. Galloway as your PI.
4. Select the following training types. If you are an undergrad, do not select the BL2+ training group.

BIOSAFETY

- ☒ Use biological materials (including microorganisms, plants, animals, recombinant or synthetic DNA/RNA, etc.), requiring BL1 or BL2 containment.
- ☐ Supervise a laboratory that uses biological materials (including microorganisms, recombinant or synthetic DNA/RNA, etc.), requiring BL1 or BL2 containment.
- ☒ Perform research with human blood or body fluids, human cells or tissues, or human cell lines or using a material containing a bloodborne pathogen (e.g. HIV, HBV, HCV, malaria).
- ☒ Conduct research requiring BL2+ containment (e.g. viral vectors that can lead to oncogenesis, propagating HIV, HBV, HCV).

CHEMICAL SAFETY

- ☒ Use potentially hazardous chemicals in a laboratory (this includes even common chemicals such as oil, solvents, paints, alcohol, acetone, etc)

SPECIALIZED SAFETY

- ☐ Work in a confined space(s) such as tanks, manholes, boilers, or shafts
- ☐ Supervise those who work in a confined space such as tanks, manholes, boilers, or shafts
- ☐ Work or supervise those that work from scaffolding, staging, lifts, powered platforms, or from any work location that presents an unprotected fall hazard of four feet or more.
- ☐ Supervise those who weld, cut or solder with a torch, braze, or grind. Supervise a designated hot works area
- ☐ Operate cranes or hoists
- ☐ Operate a forklift, battery powered pallet jack, or other material handling equipment
- ☐ Involved in planning construction or renovation activities in you department, lab, or center
- ☐ Climb a tower
- ☐ Supervise those who climb towers
- ☒ Work with cryogenic liquids (recommended training unless your DLC requires it)

5. After submitting, you will see many required trainings. Some have a required 'classroom' component, such as the *Lab Specific Chemical Hygiene training*, which will be completed with an in-lab walkaround.

Note: One of the components of the bloodborne pathogen training is the opportunity to be vaccinated for Hepatitis B or to have an antibody titer test for free.

Most of us were vaccinated for HepB as children, but that vaccine was only ~90% effective, so you may want to get the free antibody titer test. You can get a free booster or get doses of a new, more modern HepB vaccine if you no longer have HepB antibodies.

6. If you are going to be helping with mouse work, still under the 'My Profile' tab, you should click **Join Another Group** under training groups and add the **68N: Mouse** training group.

Software and webservices

Core webservices

- Create a [Quartzy account](#)⁶. Using your MIT email is recommended.
- Create a [Github account](#)⁷. You can use either a personal or MIT email.
 - Activate your student benefits by going to https://education.github.com/discount_requests/student_application. You will be asked to connect your MIT email and send in a picture of your student ID. (Because MIT does not remove emails for alums, they need to confirm active student status.)
- Create a [Zotero account](#)⁸. Using a personal email is recommended for permanence reasons.
- Create an [ORCID](#)⁹. Adding all of your active emails is recommended.

After creating these accounts, message your **Kerberos ID**, your **Github username**, your **Zotero username**, and the email you used to create your **Quartzy account** to the #iap slack channel. We'll get you added to all of the above. You will have to:

- Accept the Github invitation to the [gallowaylabmit organization](#)¹⁰.
- Accept the Zotero invitation to the [gallowaylab group](#), checking that it appears in your [group list](#)¹¹.

Shared storage

We use **OneDrive** for shared lab storage. OneDrive's web interface is slightly clunky, and occasionally you have to wait a minute for syncing to occur, but it is the MIT option that offers us the largest amount of storage space, without having bandwidth limits (unlike MIT's Google Drive).

OneDrive also has an excellent implementation of “files-on-demand”/“online sync”, where all files in the shared storage *appear* to be accessible, but do not actually take up local disk space until you open them/unless you manually trigger a download, at which point the software invisibly downloads files in the background. There's no cost to having the entire shared folder locally synced.

OneDrive also has tight integration with Office products, allowing Google Drive-esque live, multi-person editing of Office documents saved inside it.

We share presentations, software, primers, and plasmids with each other through the shared library, and all lab instruments will save data into OneDrive so everything is accessible.

When you see a file location like `instruments\data\attune` without an additional information, it is likely a path inside the OneDrive.

After being given access:

⁶ <https://www.quartzy.com/>

⁷ <https://github.com/>

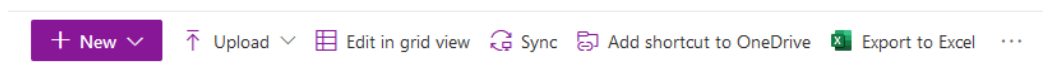
⁸ <https://www.zotero.org/user/register>

⁹ <https://orcid.org/register>

¹⁰ <https://github.com/gallowaylabmit>

¹¹ <https://www.zotero.org/groups/>

1. If not on a recent version of Windows 10 (e.g. Win 7, MacOSX, etc), download the [OneDrive client](#)¹². Recent versions of Windows come with this preinstalled.
2. Bookmark the web version here: <https://mitprod.sharepoint.com/sites/GallowayLab/Shared%20Documents>
3. On the web version, select the **Sync** button in the top tab:



4. This will trigger the OneDrive software you installed. It will ask you for a local folder to sync into. After several minutes, it will show “OneDrive is up to date”, and all files should be accessible.

Experimental software

- **SnapGene:** We use SnapGene for our cloning analysis. Download it through IST [here](#)¹³, if asked for a registration code, get it [here](#)¹⁴ (certificates required for both links).
- **FlowJo:** When initially analyzing flow cytometry data, you can install a FlowJo demo from `instruments\software`. The demo is limited time, so don't install it until you have data to analyze!
- **FIJI:** For simple image analysis, Fiji (ImageJ) gives a nice GUI interface. Download it from <https://fiji.sc>
- **CellProfiler:** CellProfiler is an excellent tool for doing image cytometry (analyzing cell-by-cell in image data). In contrast to the GUI-only tools built into the Keyence software, CellProfiler enables repeatable, pipelinable analyses. Download it from <https://cellprofiler.org/>

Other

- **Zotero:** Zotero is an excellent free, open-source citation manager. After downloading Zotero from <https://www.zotero.org/>, it should prompt you to install the Zotero Connector, a browser plugin that lets you download paper citations with one-click. If not, download the connector [here](#)¹⁵. We also have a shared Zotero group, to accumulate citations when writing manuscripts.

Several helpful plugins can be downloaded; the recommended ones are:

¹² <https://www.microsoft.com/en-us/microsoft-365/onedrive/download>

¹³ <https://downloads.mit.edu/released/snapgene/vendor-registration.html>

¹⁴ http://downloads.mit.edu/released/snapgene/group-name_registration-code.txt

¹⁵ <https://www.zotero.org/download/connectors>

Table 1: Recommended Zotero plugins

Addon name	Description
Zot-File ¹⁶	Enables useful file operations, such as extracting annotations from a marked-up PDF, transferring new papers to a tablet for annotation, and auto-file renaming.
Zu-tilo ¹⁷	Enables helpful tagging operations, such as the ability to copy/paste tags or easily add paper relationships.
Better Bib-tex ¹⁸	If you plan to use LaTeX, install this plugin before exporting to BibTeX. This addon makes nice-looking, stable citation keys that do not change on export.

Downloading Zotero plugins through Firefox

Since Zotero is built on modified Firefox, Zotero plugins appear similar to Firefox plugins. If downloading these plugins through Firefox, you will need to explicitly right click->download target; left-clicking on download links will attempt to install the Zotero plugin as a Firefox plugin, which will fail.

- **Creative Cloud:** MIT recently opened Adobe's Creative Cloud to all students. After installing the [Creative Cloud application](#)¹⁹, login with your MIT credentials, after selecting "Work/School account". You may have to wait 24 hours for activation after your first login. You should install **Acrobat** (for viewing PDFs) and **Illustrator** (for drawing vector art).
- **Inkscape:** (*Optional*) Inkscape is a free and open-source vector drawing program that can be downloaded [here](#)²⁰. Inkscape and Illustrator have many similar, but not completely overlapping features. If you have not learned to use either, pick one to start with to learn first (likely Illustrator). However, it's likely eventually worth learning both if you don't want to eventually pay for Creative Cloud. Inkscape's (Cairo) PDF import also tends to be superior, if trying to import vector images from paper PDFs.
- **Color palletes:** Having nice color-blind friendly, distinct colors to start drawings from is helpful. If making cartoon/stylized figures, the colors in-palette might be enough! If not, the palette can provide a good starting place.

You can download pre-created palettes for both [Illustrator](#) and [Inkscape](#) for the well-known Category20/20b color set, which is color-blind friendly (and becoming the default in more and more software packages):

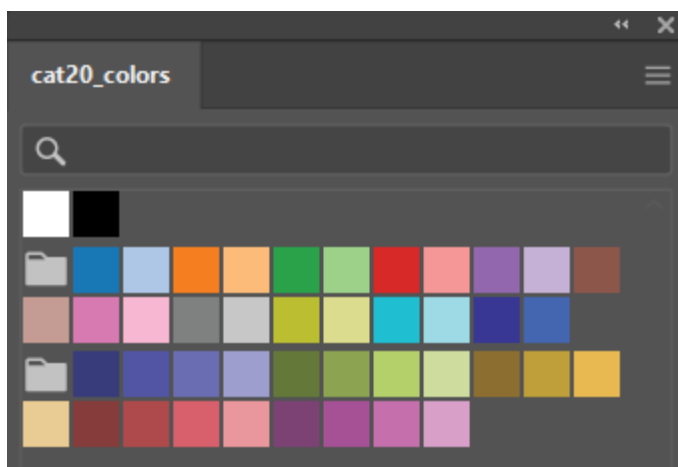
¹⁶ <http://zotfile.com/>

¹⁷ <https://github.com/wshanks/Zutilo>

¹⁸ <https://retorque.re/zotero-better-bibtex/>

¹⁹ <https://www.adobe.com/creativecloud/desktop-app.html>

²⁰ <https://inkscape.org/release/inkscape-1.0.1/>



To use these palette files, see the [Illustrator documentation](#)²¹ or the [Inkscape documentation](#)²².

Coding and collaboration

- **Slack** is how we communicate! After [downloading it](#)²³, sign into <https://gallowaylab.slack.com>. In addition to the default channels, you may want to join #sequencing to get your sequencing orders delivered right to you via Slack and join #memes for obvious reasons.
- **Git:** For any code/code-like thing (LaTeX, other plain-text files) you write, Git is the standard way to share and collaborate with others in addition to tracking your entire version history.

You must install the base command-line tools from [here](#)²⁴, make sure that you select your operating system and not the “Download source code” button!

Tip: When installing Git, you may want to change Git’s default editor to something other than Vim, such as VS Code.

When asked about adjusting the PATH environment, choose the **Git from the command line and also from 3rd-party software** option; this makes sure all the other software also has Git access. All other defaults are fine, but can be changed if you want.

After installation, you should set your global identity on that computer (e.g. what name/email gets stored alongside the work you do).

Open a terminal (Terminal on MacOS, Powershell on Windows) and type the following lines (without the beginning \$, which identifies here that we are typing this into a terminal), substituting your name and email (giving an email you associated with your Github account).

²¹ https://helpx.adobe.com/illustrator/using/using-creating-swatches.html#share_swatches_between_applications

²² <https://inkscape-manuals.readthedocs.io/en/latest/palette.html>

²³ <https://slack.com/downloads>

²⁴ <https://git-scm.com/downloads>

```
$ git config --global user.name "Full Name"
$ git config --global user.email email_address@example.com
```

- **Github Desktop:** This program is a good basic GUI Git tool, in case the command line interface/built in editor interfaces aren't for you. Download it [here](#)²⁵.
- **Python:** Python is an excellent “Jack of all trades” language; we use it extensively. If you are on MacOS, you may have Python3 pre-installed; you can check by typing `python3` at a terminal. If you do not have Python preinstalled, you should download it from [here](#)²⁶. Click the latest version download from the top, then scroll down and select the 64-bit installer for your OS.

When installing, select **Add Python to PATH**; this ensures that when you type `python` at a terminal, you get this version you just installed. Other software can also access this “default” installation.

On snakes and Anaconda

If you have Anaconda installed and don't have an explicit reason to need it (e.g. conda-only packages), it is recommended to uninstall Anaconda and install Python directly this way.

With modern Python, the benefits that Anaconda initially brought to the field (virtual environments and pre-compiled packages) are now integrated into the normal Python ecosystem, making Anaconda unnecessary. We also don't want multiple Python versions competing.

After installing, to make sure it worked, open a new terminal and type the following, checking that the output looks similar to the following.

```
$ python
Python 3.9.1 (tags/v3.9.1) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
```

Exit the Python prompt by typing `exit()`

After exiting Python, you can install the normal “semi-base” packages needed for most data analysis:

```
$ pip install numpy scipy pandas matplotlib jupyter jupyterlab
```

- **R:** Many bioinformatic tools are written in R, so we also use R. From [here](#)²⁷, download the main package (MacOS) or both the `base` entry and the `Rtools` entry (Windows).
- **VSCode:** (*Optional*) Having a good *plain-text editor* (e.g., not Word) is important for coding, and is ultimately up to personal taste. If you have your own favorite, feel free to not install VS Code. If you are used to language-specific IDE's like MATLAB, IDLE, or RStudio, VS Code allows you to do editing, debugging, previewing, source control, etc in a mostly language-

²⁵ <https://desktop.github.com/>

²⁶ <https://www.python.org/downloads/>

²⁷ <http://lib.stat.cmu.edu/R/CRAN/>

agnostic manner; once you customize it to how you want, you can use it for all of your coding.

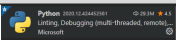
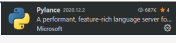
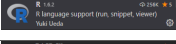
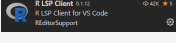
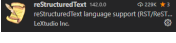
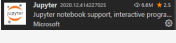
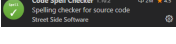
The recommended editor is VS Code, downloadable [here](https://code.visualstudio.com/)²⁸.

After installing, you should click the extensions button:



and search and install the following extensions (type in the name, click the install button).

Table 2: Recommended VS Code extensions

Addon name	Image	Description
Python		Enables Python debugging, running, and IntelliSense (in-line help while typing).
Pylance		Faster 'language server' for Python, which means the IntelliSense is faster and more accurate.
R		Base language support for R.
R LSP Client		The VS Code side of the R language server. Before installing this, run <code>install.packages("languageserver")</code> inside an R prompt.
reStructured-Text		Enables reStructuredText support, the language used to write this documentation, among others.
Jupyter		Inline Jupyter notebook support. No more need to launch Jupyter in a web browser, just do it inside VS Code!
Code Spell Checker		Inline spell checker that is intelligent enough to not flag specific language-specific words, but still can spell check comments and variable names.

- **RStudio:** (Optional) If you don't feel like using VS Code for your R work, the excellent, well-polished standard IDE is RStudio Desktop, downloadable [here](https://rstudio.com/products/rstudio/download/#download)²⁹.

²⁸ <https://code.visualstudio.com/>

²⁹ <https://rstudio.com/products/rstudio/download/#download>

4.1.3 An introduction to Git

Motivation

Even if you haven't used version control software before, you have probably encountered two types of version control:

1. Linear time-based snapshots. This can either be manually done (e.g. you append a date to documents and copy files when you want to make a new version), or automatically done (e.g. you are storing documents in Dropbox, and it stores time snapshots for the last 30 days automatically).
2. Online, live multi-person editing, like in Overleaf and Google Drive.

These techniques are useful, but have limitations. Time-based snapshots often fail if you want to have multiple people editing simultaneously without massive headaches, and online live editing only works if you have permanent internet access, intermediate states are meaningful (e.g. saving mid-sentence is fine for manuscripts, but would throw a syntax error for code), and you don't actually care about saving a detailed history.

For us, having robust version control without these limitations makes our coding work easier to share, easier to reproduce (how do you regenerate a plot made with a previous code version you didn't know you needed?), and easier to do shared work.

In the CS community, this is a solved problem; the field has coalesced around using the version control system called `git`. There are competing version control systems, including Subversion, Mercurial, and Fossil, but `git` has emerged as the clear winner with the emergence of supporting webservices Github, Sourceforge, GitLab, and others.

Unfortunately for us, `git` is also by far the most powerful and complicated version control system, so before jumping into `git` commands, let's consider an example:

Git is a simple, but extremely powerful system. Most people try to teach Git by demonstrating a few dozen commands and then yelling "tadaaaaa." I believe this method is flawed. Such a treatment may leave you with the ability to use Git to perform simple tasks, but the Git commands will still feel like magical incantations. Doing anything out of the ordinary will be terrifying. Until you understand the concepts upon which Git is built, you'll feel like a stranger in a foreign land.

—Tom Preston-Werner, [The Git Parable](https://tom.preston-werner.com/2009/05/19/the-git-parable.html)³⁰

Say you are trying to do version control for files related to a paper manuscript. You have some figures, and a main `.docx`

```
.
├── figure1.ai
├── figure2.ai
└── manuscript.docx
```

How should you track the version history of this? First create a new directory/folder, which is called `working` here, as it is the directory you would be actively working on/editing.

³⁰ <https://tom.preston-werner.com/2009/05/19/the-git-parable.html>

```
.
└─ working
    ├── figure1.ai
    ├── figure2.ai
    └─ manuscript.docx
```

After some hacking in Illustrator, you're happy with the way the figures look. To save this version, you just copy the entire working folder to a `snapshot-01` folder and promise yourself that you won't further edit what is inside the snapshot folders.

```
.
├─ snapshot-01
│   ├── figure1.ai
│   ├── figure2.ai
│   └─ manuscript.docx
└─ working
    ├── figure1.ai
    ├── figure2.ai
    └─ manuscript.docx
```

After you do some more work, (some directories are shown collapsed, without showing content), such as adding new figures, you might have lots of snapshots:

```
.
├─ snapshot-01
├─ snapshot-02
├─ snapshot-03
├─ snapshot-04
│   ├── figure1.ai
│   ├── figure2.ai
│   ├── figure3.ai
│   └─ manuscript.docx
└─ working
    ├── figure1.ai
    ├── figure2.ai
    ├── figure3.ai
    └─ manuscript.docx
```

How are you supposed to remember what we changed in each of these? To help yourself, you decide to add a new file, `message.txt` to each snapshot created. Inside the message, you decide to add the date, our name, and some free-form message that describes what happened in that version snapshot. A `message.txt` might look like:

```
Author: Christopher Johnstone <cjohnsto@mit.edu>
Date:   Jan 11 2021

    Made Figure 3 300% more aesthetic!
```

The snapshots now look like the following. Note that we add the `message.txt` after we copy the working folder to a new snapshot folder.

```

.
├── snapshot-01
├── snapshot-02
├── snapshot-03
├── snapshot-04
├── snapshot-05
│   ├── message.txt
│   ├── figure1.ai
│   ├── figure2.ai
│   ├── figure3.ai
│   └── manuscript.docx
└── working
    ├── figure1.ai
    ├── figure2.ai
    ├── figure3.ai
    └── manuscript.docx

```

After some more work, you end up revising the manuscript while also trying out a new revision of `figure1.ai`. You want to save your updated manuscript version, but the figure isn't quite ready yet. How do we save the manuscript edits into a snapshot while leaving the in-progress figure edits out of the snapshot? There isn't an obvious way to do this with this folder setup, so you decide to make another special directory called `stage` (as in a stage to set and later snapshot).

With the edits marked with a `*`, we initially have:

```

.
├── snapshot-01
├── snapshot-02
├── snapshot-03
├── snapshot-04
├── snapshot-05
│   ├── message.txt
│   ├── figure1.ai
│   ├── figure2.ai
│   ├── figure3.ai
│   └── manuscript.docx
├── stage
│   ├── figure1.ai
│   ├── figure2.ai
│   ├── figure3.ai
│   └── manuscript.docx
└── working
    ├── *figure1.ai
    ├── figure2.ai
    ├── figure3.ai
    └── *manuscript.docx

```

Now instead of directly copying the `working` folder into a snapshot, we first copy the changes we want into `stage`, while leaving files we are still actively editing in `working`. Copying the modified manuscript to `stage`:

```
.
├── snapshot-01
├── snapshot-02
├── snapshot-03
├── snapshot-04
├── snapshot-05
│   ├── message.txt
│   ├── figure1.ai
│   ├── figure2.ai
│   ├── figure3.ai
│   └── manuscript.docx
├── stage
│   ├── figure1.ai
│   ├── figure2.ai
│   ├── figure3.ai
│   └── *manuscript.docx
└── working
    ├── *figure1.ai
    ├── figure2.ai
    ├── figure3.ai
    └── *manuscript.docx
```

then we make a new snapshot from stage:

```
.
├── snapshot-01
├── snapshot-02
├── snapshot-03
├── snapshot-04
├── snapshot-05
│   ├── message.txt
│   ├── figure1.ai
│   ├── figure2.ai
│   ├── figure3.ai
│   └── manuscript.docx
├── snapshot-06
│   ├── message.txt
│   ├── figure1.ai
│   ├── figure2.ai
│   ├── figure3.ai
│   └── *manuscript.docx
├── stage
│   ├── figure1.ai
│   ├── figure2.ai
│   ├── figure3.ai
│   └── *manuscript.docx
└── working
    ├── *figure1.ai
    ├── figure2.ai
    ├── figure3.ai
    └── *manuscript.docx
```

Finally, this whole setup seems to be working well, and you want to go share your work with fellow

lab-mates. While you could just copy the entire folder and share created snapshots, this would be inherently linear; only one person could work on this at a certain time! To solve these issues, among others, you decide to assign arbitrary, unique names to snapshots, such as `2a8aba`. With these arbitrary names, it's hard to tell what order you made the snapshots in, so you also add a `parent` field.

```
.
├── snapshots
│   ├── c3612a
│   ├── 86ac2d
│   ├── acd748
│   ├── fb9742
│   └── 12d276
│       ├── message.txt
│       ├── figure1.ai
│       ├── figure2.ai
│       ├── figure3.ai
│       └── manuscript.docx
├── 2a8aba
│   ├── message.txt
│   ├── figure1.ai
│   ├── figure2.ai
│   ├── figure3.ai
│   └── *manuscript.docx
├── stage
│   ├── figure1.ai
│   ├── figure2.ai
│   ├── figure3.ai
│   └── *manuscript.docx
└── working
    ├── *figure1.ai
    ├── figure2.ai
    ├── figure3.ai
    └── *manuscript.docx
```

After this renaming, the most recent `message.txt` (in snapshot-06, renamed to `2a8aba`) could read:

```
Snapshot: 2a8aba
Parent: 12d276
Author: Christopher Johnstone <cjohnsto@mit.edu>
Date: Jan 12 2021

Updated the manuscript with method details.
```

We're almost ready to jump into Git; the last thing to note from this example is that specifying parents of snapshots means that we no longer (necessarily) have a linear history; instead, the history forms a tree (a graph without cycles). Trees have nice properties, like always being able to determine all of your ancestors.

Git by example

Heavily inspired by the [Software Carpentry Git lesson](https://swcarpentry.github.io/git-novice)³¹, we will start you off learning Git by editing the very repository we are working on!

Note: There is a

```
$ pip install sphinx sphinx-rtd-theme sphinx-last-updated-by-git
```

³¹ <https://swcarpentry.github.io/git-novice>

4.1.4 Software Tips and Tricks

One date format to rule them all/naming

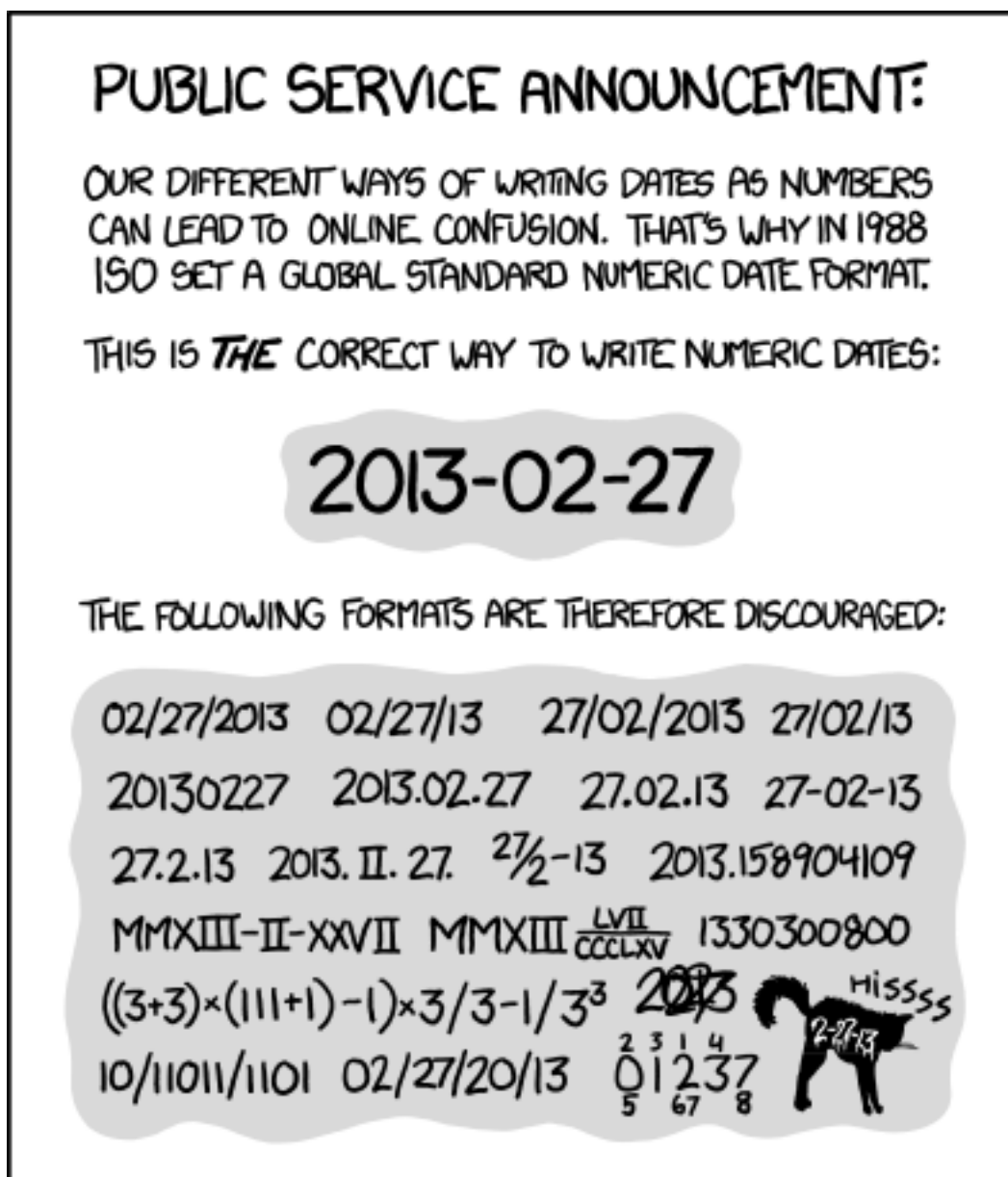


Fig. 1: Courtesy of [xkcd](https://xkcd.com/1179/)³³

In file naming schemes, we typically use the YYYY-MM-DD format. The separators don't typically matter, you could use dashes or underscores or periods.

This date format is both unambiguous and also sorts well (e.g. a lexicographic/alphabetical sort sorts in the correct time order).


³² https://imgs.xkcd.com/comics/iso_8601.png

³³ <https://xkcd.com/1179/>

Finally, avoid using spaces in file and directory names! It's not a problem for most code, but it's just a good best practice that makes typing these paths in easier in a terminal.

Paper RSS feeds

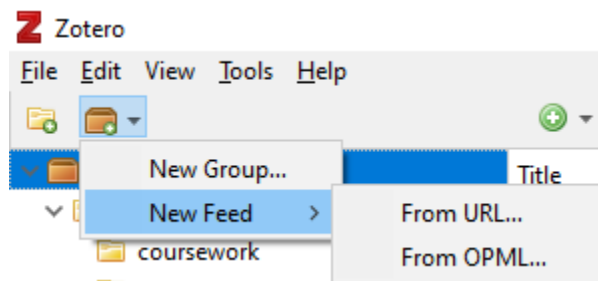
Staying on top of the firehose of papers can be difficult. Using a RSS (Really Simple Syndication) aggregator can be helpful. RSS is a relatively old technology that allows for websites to push lists of content over time; the first podcasts were actually syndicated/released using RSS.

Most journals have RSS feeds; you can find it by searching for a journal name + `rss`, or by looking around with the RSS icon: 

For example, the main Nature feed lives at <http://feeds.nature.com/nature/rss/current>.

After finding some feeds you are interested in, you likely want to use a **feed aggregator**, something that combines all of the new pushed papers into a single feed. A very popular feed aggregator is [Feedly](#)³⁴. It has a website and mobile apps. The free version is more than sufficient for most purposes; it lets you combine up to 100 RSS feeds into 3 separate feeds.

You can also view feeds directly in Zotero! If you export your feed list from Feedly as a OPML file, you can also direct-import it into Zotero.



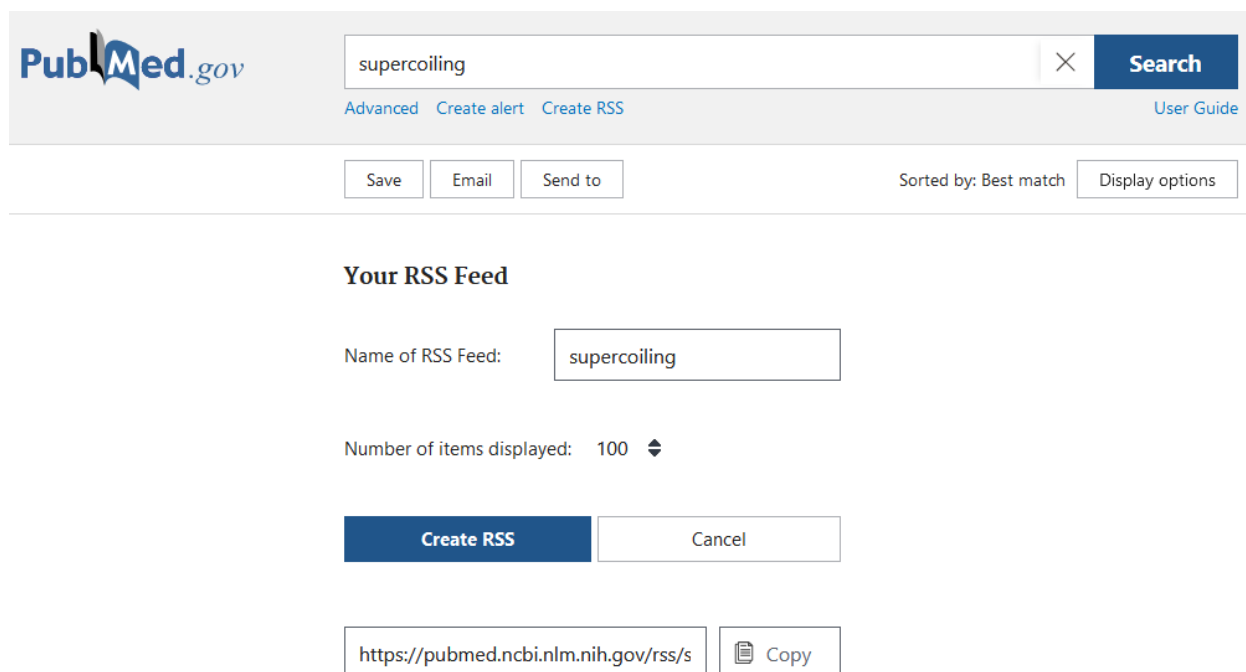
Creating feeds from searches

Sometimes you want to have a feed of papers matching some search terms. You can easily do this with Pubmed and ProQuest; whenever a new paper that matches your search terms gets added, it will get added to your RSS feed aggregator. Unfortunately, Google Scholar does not allow you to create RSS feeds (this would easily enable competition with their services).

For Pubmed searches, start at <https://pubmed.gov>, and design your search. You may want to be pickier; a search for just `p53` is going to return a lot of junk!

After you have a search of your liking, click the `Create RSS` button, and bump up the returned number of items to 100. Then, you can directly use that feed URL in Feedly or Zotero!

³⁴ <https://www.feedly.com/>



PubMed.gov

supercoiling

Advanced Create alert Create RSS User Guide

Save Email Send to

Sorted by: Best match Display options

Your RSS Feed

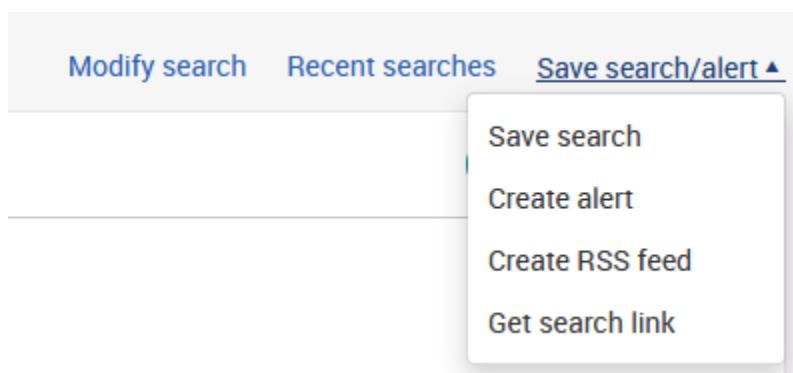
Name of RSS Feed: supercoiling

Number of items displayed: 100

Create RSS Cancel

https://pubmed.ncbi.nlm.nih.gov/rss/s Copy

To add a Proquest feed, you can go to <https://search.proquest.com>, make a search, then click the “Save search/alert” button to create a RSS feed:

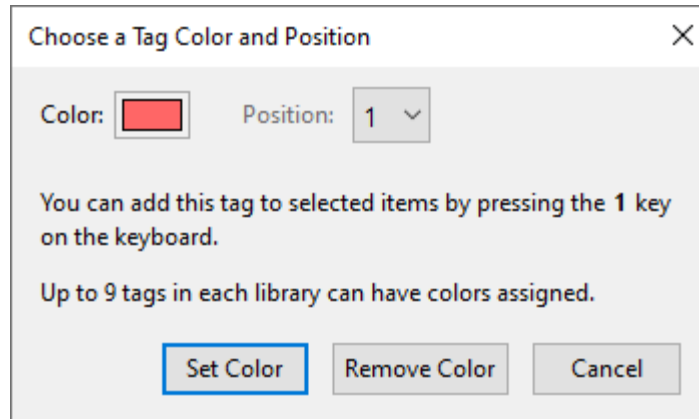


However, because Proquest is a MIT-sponsored database, you may have trouble accessing feed items outside of the MIT network.

Zotero

There are several built-in features of Zotero that make paper reading more efficient.

1. **Color tags:** Zotero sometimes adds automatic tags based on article metadata, but you can also add tags of your own under the details menu. In addition, you can select up to 9 tags at any one time as quick/color tags. In the tag menu in the bottom left, right clicking on a tag lets you assign a number and a color; tagged entries will have small colored squares viewable at a glance.



2. **Notes:** When you add notes through the sidebar, they become fully searchable! You can also embed images and other content within notes. This is particularly helpful for summarizing papers; I use the following note template:

Top line summary:

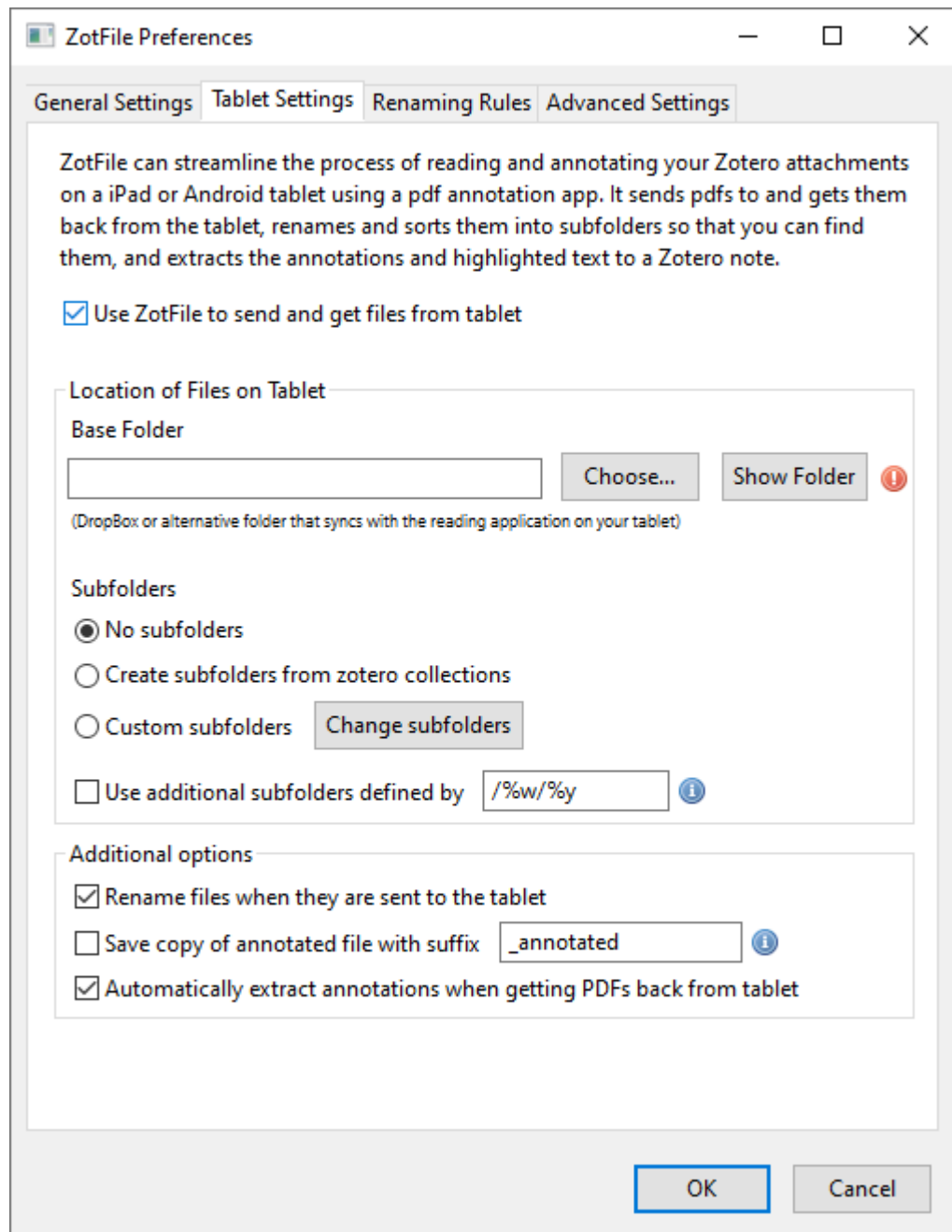
What they did/highlights:

Why do we care:

Limitations:

Extensions:

3. **Full-text searching:** The search bar has several available options through the dropdown. Switching to “everything” includes full-text searching of linked PDFs (e.g. it will look for phrases inside every linked PDF)!
4. **Tablet syncing:** If you install the Zotfile extension, you can setup automatic transfer and syncing with a tablet for annotation purposes. You will need some cloud syncing application (Dropbox, OneDrive, Google Drive) that is accessible both on your computer through your annotation app on your tablet. Point ZotFile at this folder, and it will automatically check for updates and pull the annotated PDFs back into Zotero.



Better Quartzzy

While Quartzzy is great for inventory purposes, and the interface for the plasmid database isn't too bad, the web-interface leaves a lot to be desired. By default, you can't really read the plasmid names even after you drag the CAS # field to the second position:

<input type="checkbox"/> ITEM NAME ▾	CAS #	VENDOR	AMOUNT	LOCATION	TYPE	OWNER	ADDED	UPDATED	
<input type="checkbox"/> pKG521	pENTR...				Bacteri...	Adam Bel...	Jan 5, 2021	Jan 5, 2021	Request
<input type="checkbox"/> pKG520	pMXs-...				Bacteri...	Adam Bel...	Dec 15, 2020	Dec 15, 2020	Request
<input type="checkbox"/> pKG519	pCW5...				Bacteri...	Adam Bel...	Dec 15, 2020	Dec 15, 2020	Request
<input type="checkbox"/> pKG518	phage...				Bacteri...	Adam Bel...	Dec 15, 2020	Dec 15, 2020	Request
<input type="checkbox"/> pKG517	pENTR...				Bacteri...	Adam Bel...	Dec 15, 2020	Dec 15, 2020	Request

To fix this, there is a Quartzzy enhancer userscript that makes it look like the following!

<input type="checkbox"/> ITEM NAME ▾	CAS #	VENDOR	AMOU...	LOCATI...	TYPE	OWNER	ADDED	UPDAT...
<input type="checkbox"/> pKG521	pENTR-MCP-mCherry-NLS (K)				Bacte...	Adam B...	Jan 5, 2021	Jan 5, 2021
<input type="checkbox"/> pKG520	pMXs-ERKKTR-Clover (A)				Bacte...	Adam B...	Dec 15, 2020	Dec 15, 2020
<input type="checkbox"/> pKG519	pCW57-mCherry-MCP (A)				Bacte...	Adam B...	Dec 15, 2020	Dec 15, 2020
<input type="checkbox"/> pKG518	phage-mCherry-MCP (A)				Bacte...	Adam B...	Dec 15, 2020	Dec 15, 2020
<input type="checkbox"/> pKG517	pENTR-IsII-Halo (K)				Bacte...	Adam B...	Dec 15, 2020	Dec 15, 2020

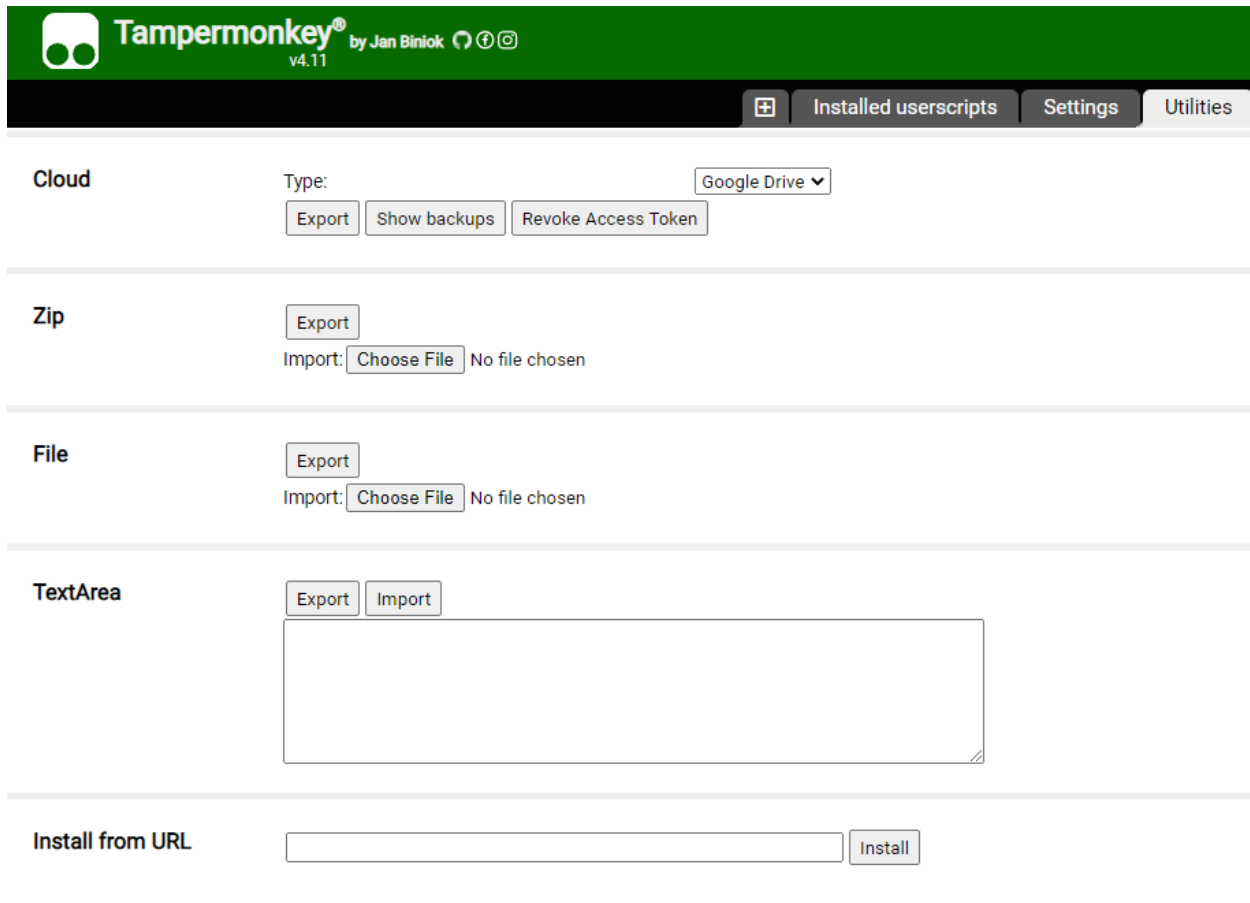
The plasmid field is larger, and the plasmid resistance (Amp/Kan/Chlor) is listed directly below the plasmid.

This feature is implemented using something called **userscripts**; these are small Javascript scripts that get injected into webpages; effectively they are mini browser extensions.

First, you should install a userscript manager like [Tampermonkey](https://www.tampermonkey.net/)³⁵.

Then, you should be able to click on this link to add the userscript: https://gist.github.com/meson800/f28e64d532da9b0fe2a1d22480ea5cda/raw/quartzy_enhancer.user.js

or in the Tampermonkey Utilities tab, you can use the **install from URL** option:



The screenshot shows the Tampermonkey v4.11 interface. The top bar is green with the Tampermonkey logo and version number. Below it is a navigation bar with tabs: '+', 'Installed userscripts', 'Settings', and 'Utilities'. The 'Utilities' tab is selected. The main content area is divided into sections: 'Cloud', 'Zip', 'File', 'TextArea', and 'Install from URL'. The 'Cloud' section has a 'Type:' dropdown set to 'Google Drive' and buttons for 'Export', 'Show backups', and 'Revoke Access Token'. The 'Zip' section has an 'Export' button and an 'Import' section with a 'Choose File' button and 'No file chosen' text. The 'File' section has an 'Export' button and an 'Import' section with a 'Choose File' button and 'No file chosen' text. The 'TextArea' section has 'Export' and 'Import' buttons and a large text area. The 'Install from URL' section has a text input field and an 'Install' button.

Regex help



If you ever need help debugging or designing a regular expression, try using <https://regex101.com/>

³⁵ <https://www.tampermonkey.net/>

Fonts

Helvetica Neue is a good sans-serif font that is based on everyone's favorite font, Helvetica. Download it [here](#).

For a good monospaced/code/terminal font, [Fira Code](#)³⁶ is excellent. Besides looking nice, Fira Code has something called **font ligatures**. These are originally defined for special letter combinations, like æ for adjacent æ. In Fira Code, common programming combinations are given special ligature symbols that appear as you type normally. You often have to enable ligatures in the editor you are using.

	Fira Code v5	Fira Mono
	ligatures:  ON	ligatures:  NO
Common		
Arithmetics	++ -- /= && =	++ -- /= && =
Scope	-> => :: __	-> => :: __
Equality	= ≡ ≠ ≠= == === ≠ ≠=	= == != /= == == != !==
Comparisons	≤ ≥ ≤ ≥ ⇔	<= >= <= >= <=>

³⁶ <https://github.com/tonsky/FiraCode/releases>

4.1.5 On Terminals and Shells

Motivation

Knowing the basics of using a command line interface is occasionally helpful. While a lot of software has nice GUI interfaces, making these interfaces takes a large amount of work. Software written by a small number of people (like research groups!) often are only accessible via a CLI. In many cases, a command line interface can also be a faster way to do certain operations.

Even if you avoid using a terminal in daily computing, a terminal is often the only way that you can access high performance computing clusters.

The basics

Lots of words get thrown around: CLI, terminals, bash, shells, command prompts. For these purposes:

- A **terminal** is the program that runs on your computer and handles all of the low-level input output details. It is responsible for drawing to the screen, getting keyboard input, handling the clipboard, selecting fonts, and so on.

Common terminals might be the programs *Terminal* or *iTerm2* on MacOS or *Windows Terminal*, *Powershell*, and *Command prompt* on Windows (more on this later).

- A **shell** is a command-line program (e.g. instead of interacting with us through a graphical interface, you type stuff in line by line) that lets the user take actions like modify files, view program output, run other command-line programs, and so on.

Inside a terminal, you run a shell. Somewhat confusingly, the default terminals on both MacOS and Windows run a default shell, so that the shell and terminal appear identical.

Common Unix shells (runnable on MacOS and Linux) are the original shell, `sh`, along with `bash` (common default on Linux), `zsh` (recent default on MacOS), and others like `csh` and `dash`. The two major Windows shells are `cmd` (old-school, going back to DOS) and `powershell` (Windows' modern shell).

When a shell starts it, it displays a **prompt**, showing that it is ready for input. Once you are done typing in your command, you hit enter to run that command.

When you run a command line program, the shell will show the prompt again when it's ready for more input. In these notes, we will use `$` to represent generic prompts (and `>` for the default Powershell prompt); lines without the `$` are example output of what you will see when you enter the command. When typing in commands, you should *not* type in the dollar sign! Your specific shell may display a more complicated prompt than a dollar sign, such as showing the current directory that you are currently in.

An example is:

```
$ echo "test"
test
```

³⁸ MacOS image from <https://ohmyz.sh/>



Fig. 2: An example of the difference between terminals and shells³⁸

If we have an operating-system specific command to show, we'll show them in specific boxes, and we'll use the Powershell prompt symbol `>` for the Powershell examples.

```
$ echo "test"
test
```

```
> echo "test"
test
```

Finally, throughout this we will talk about **files** and **directories**. Directories are more commonly known as **folders**, but **directory** is the common shell terminology, so will be used here for consistency. However, saying folder is totally fine unless someone is being *really* pedantic.

Commands

In a shell, you enter commands to run them. When processing a line, the line you entered is first separated by the spaces present. Consecutive spaces are treated as a single space, with text in quotes being represented as a single “word”.

Then, the first word is taken as the command name, and looked up in the list of installed command-line programs (e.g. see if it is present in `PATH`). The rest of the line is given to that program as it's **command line arguments**.

By convention, command line arguments that start with dashes are normally **options**. By convention, these options typically either have long names and start with two dashes, or have a “short-hand” form with a single dash and a single letter. Arguments that don't start with dashes are typically user-specified.

For example, in the following commands:

```
$ git commit --message "Testing out a commit"
$ git commit -m "Testing out a commit"
```

the command `git` is called in both cases, and is given the argument list:

(`commit, --message, Testing out a commit`) in the first case or (`commit, -m, Testing out a commit`) in the second. These two calls are identical in this case, because `-m` is shorthand for `--message`.

Demo

For the command line commands:

```
$ python -m venv env
$ git add testing.py module.py "explanation revised.docx"
```

what command is called in each case? What arguments are given to that command?

In the first example, the program `python` is ran with arguments (`-m, venv, env`).

In the second example, the program `git` is ran with arguments (`add, testing.py, module.py, explanation revised.docx`).

Interface basics

While the shell is minimalistic, there are three features that make our life easier:

1. **Job control:** Pressing Control and C (notated as `Control-C`, `Ctrl-C`, or `^C`) does **not** copy inside a terminal. Instead, `^C` is the command to quit the actively running program. The program is given a chance to clean up after itself (e.g. this is like hitting the exit button in a program, not force-closing it).

On Unix-derived systems (Linux and MacOS), you can additionally pause a running program by pressing `Ctrl-Z`. When you pause the program, you will see `[1]+ Stopped` and you will be back at the shell prompt. To resume the program, type `fg` (to bring the program back to the foreground).

2. **Clipboard:** Because control-C does not copy, we need some other way of using the clipboard. On Mac, this is easy; copy and paste are typically bound to Command-C and Command-V. On Windows and Linux, it depends on what terminal you are using. A typical copy/paste solution binds the keyboard to right-click. On standard Powershell, you can select a region with your mouse and press enter to copy that to the clipboard. To paste, right click inside the terminal region.
3. **Tab completion:** Typing out full file names gets tiring, especially when you have long file names or deeply nested directory structures. Tab completion saves us: if you have a filename partially written, hitting Tab attempts to auto-fill the rest of the name. If there is a single unique file that matches what you have typed so far, that name is filled. If there are multiple files that might match, then the exact behavior differs per shell. Bash and similar shells will typically complete as much of the name as possible, but then stop. If you double-tap Tab in Bash, it will print a list of all possible matching files. In Powershell, pressing Tab cycles between files that match.
4. **History:** Retyping common commands also gets tiresome. You can access your command line history (e.g. the previous lines you have typed) by pressing the up and down arrows.

Starting off at home

When you first open a terminal, your shell will likely start off in your **home directory**, also known in shorthand as `~`. Each user has its own home directory. All of the user directories that you are used to accessing through Windows Explorer or Finder, such as `Desktop`, `Downloads`, or `Documents` are subdirectories of your home directory.

The actual location of your home directory differs, but is typically something like `C:\Users\Username` on Windows, `/users/Username` on MacOS, and `/home/Username` on most Linuxes.

So that we don't have to type that large thing every time, `~` is short-hand notation for whatever your home directory is. That is, the location of your downloads folder could be written either as `C:\Users\Username\Downloads` or more simply as `~\Downloads`

Note: You may have noticed earlier that these directory paths have been written differently between the two operating systems. In short, due to backwards compatibility, Windows uses the backslash `\` as the path separator (written between directory names), whereas all Unix-derived operating systems including MacOS and Android use the forward slash `/` as the path separator.

Most of the time you can just use the forward slash without worry; `powershell` on Windows will auto-convert from forward slashes to backslashes if you use forward slashes, but when programming you should keep this in mind and not manually use slashes when constructing paths to filenames. It still may work, but you should ideally use filesystem-aware techniques, like using `os.path` or `pathlib` in Python.

The shell has a current location; imagine it as having a Finder/Explorer window open to some directory on your computer. This current location is called the (current) **working directory**. This is How do we know what directory we are in while using the shell? Our first command we will learn is `pwd`:

Command: `pwd`

`pwd` stands for **print working directory**, and does just that; it tells you what your current location is, in full detail (e.g. the entire path, not in shorthand). If you ever get lost, just type `pwd`!

This output is similar across operating systems; it is a little more verbose in Powershell.

Example output right after launch, so that you are starting in your home directory:

```
$ pwd
/users/username
```

```
> pwd

Path
----
C:\Users\Username
```

If we want to know what is inside the current directory, we can use `ls`:

Command: `ls`

`ls` stands for **list**, and lists every file and directory inside the current working directory.

If you were to run it in your home directory, you might get something like:

```
$ ls
Desktop    Downloads  Pictures
Documents  Music      Videos
```

If you want to see what is inside one of these directories, `ls` takes command line arguments specifying which directory you'd like the view:

```
$ ls Documents
10-50      10-40      10-34
research
```

To view **hidden files** (on MacOS/Linux, these are files/directories that start with a period; on Windows, these are files/directories with a hidden attribute set), we need to pass `ls` a command line option. This differs between shells, but on bash/zsh/etc, you use `--all` or `-a` to show hidden files as well:

```
$ ls -a
.      Desktop  Music    Videos
..     Documents Pictures
.bashrc Downloads .profile
```

In Powershell, we pass the option `-Force`:

```
> ls -Force
Directory: C:\Users\username

Mode                LastWriteTime         Length Name
----                -
d--h--             1/11/2021   7:19 PM           .git
d-----            1/11/2021   7:19 PM        Desktop
d-----            1/11/2021   7:19 PM       Documents
d-----            1/11/2021   7:19 PM      Downloads
d-----            1/11/2021   7:19 PM        Music
d-----            1/11/2021   7:19 PM       Pictures
d-----            1/11/2021   7:19 PM       Videos
```

Moving away from home

To move what directory we are in, we can use `cd`:

Command: `cd`

`cd` stands for **change directory**, and switches the current working directory to whatever directory you give it. This is the major way that you move around the various directories to find files.

```
$ pwd           # Start off in your home directory
/Users/username/
$ cd Downloads  # move into the Downloads directory
$ pwd
/Users/username/Downloads
$ cd ~          # return the the home directory
$ pwd
/Users/username
```


Relative and absolute paths

The earlier examples have hinted at the existence of two types of paths/ways to reference files.

The first is using an **absolute path**; this is what we call specifying the entire path from the filesystem “root” to the file of interest. On Windows, this means paths like `C:\Users\username\Downloads`, where we specify the drive followed by every path component.

On MacOS and Linux, absolute paths start at the root, which is the special name given to the path `/`, so absolute paths look like `/Users/username/Downloads`.

In contrast, **relative paths** allow you to more concisely reference files and directories, as the paths are calculated relative to the current working directory.

It is fairly intuitive how this works for going into subdirectories; just specify the subdirectory name. To be able to reference directories “above” yourself in the tree, we need some way to reference these parent directories.

Luckily this is standardized; there are two special pseudo-directories accessible everywhere on the filesystem; the ‘current directory’ `.` and the ‘parent directory’ `..`. The current directory is always a sort of empty operation, but is useful if you want to run scripts in the same directory as yourself.

When these are passed to a command, they are evaluated starting at the current working directory.

Say that we start off in our downloads directory, `/Users/username/Downloads`. Then changing directory to relative directory `..` means going one step “up”, to `/Users/username`

```
$ pwd
/Users/username/Downloads
$ cd ..
$ pwd
/Users/username
```

We can go up multiple layers at a time by combining these pseudo-directories together. For example, to go up two directories to `/Users` from `/Users/username/Downloads`, you could just write `cd ../../`.

You can actually combine absolute and relative paths; the parent directory `..` will always go “up” a directory, effectively removing what comes to the left if combined in this way. For example, the paths `/Users/username` and `/Users/username/Desktop/..` both point to the same thing.

Demo

If your shell starts in the Downloads directory `/Users/amanda/Downloads`, which of the following will navigate to the directory `/Users/amanda/data`? `/Users/amanda` is your home directory.³⁹

1. `cd .`
2. `cd /`
3. `cd /Users/amanda/data`

³⁹ Inspired by the [Software Carpentry examples](#)⁴⁰, license CC-BY-4.0

⁴⁰ <https://swcarpentry.github.io/shell-novice/02-filedir/index.html>

4. `cd ../../`
5. `cd home/data`
6. `cd ../data`
7. `cd ~/data`

The 3rd, 6th, and 7th examples will navigate to the proper directory.

1. `.` will stay in the same directory, `/Users/amanda/Downloads`
 2. `/` is an absolute path to the filesystem root, not the correct directory.
 3. `/Users/amanda/data` is the full absolute path to the desired directory, so this works.
 4. `../../` evaluates to `/Users`, the wrong directory.
 5. `home/data` will give an error, as it tries to navigate to `/Users/amanda/Downloads/home/data`
 6. `../data` evaluates to `/Users/amanda/Downloads`, the correct path.
 7. `~/data` also works, as `~` expands to `/Users/amanda`
-

File operations

Now that we can navigate around, we can learn file operations. The first is conceptually the simplest, as it creates a new directory:

Command: `mkdir`

`mkdir` stands for **make directory**. It creates a directory name equal to that of the argument it gets passed.

```
$ pwd           # Start off in the home directory
/Users/username/
$ cd test      # try moving into the test directory; it fails!
cd: test: No such file or directory
$ mkdir test   # Create the test directory
$ cd test      # Now the cd succeeds
```

Importantly, `mkdir` on Linux/MacOS can only make a single directory by default, so an error will occur if you try to create nested directories in one command (e.g. if we want to create the directories `test/inner_test` without first creating `test`, we'll get an error). Powershell does not have this limitation.

If we do want to create multiple nested directories, we can use the `-p` or `--parent` flag to tell `mkdir` that it is allowed to create parent directories if they don't exist.

Now that we can create directories/folders, how do we actually move files around? Using `mv`!

Command: mv

`mv` stands for **move**, and takes at least two arguments. We use `mv` to both move and rename files (renaming is just moving!).

The input arguments are `mv <source> <destination>`. Source can be a single file/directory, or it can be multiple! However, if multiple source files are given, then the given destination **must be a directory!** Put another way, even though there can be multiple source files/directories, there can only be one destination.

If we want to rename a file we can do so with move:

```
$ ls
test.txt
$ mv test.txt old_test.txt
$ ls
old_test.txt
```

We can also rename directories in the same way:

```
$ mkdir test_dir
$ ls
test_dir
$ mv test_dir new_dir
$ ls
new_dir
```

If we want to move files, we can move them by naming them one by one. Consider the case where we have several `.txt` files in a subdirectory. We can use a wildcard to move everything matching a certain wildcard pattern:

```
$ ls
inner_dir
$ cd inner_dir
$ ls
test1.txt test2.txt test3.txt
$ mv test*.txt ../ # Move files to the parent directory (where we started)
$ ls              # now there are no more files in inner_dir
$ cd ..
$ ls              # but they are in the parent directory!
inner_dir test1.txt test2.txt test3.txt
```

Copying files is very similar to moving files. In fact, it has almost all of the same semantics: the one difference is in how you have to copy directories.

Command: cp

`cp` stands for **copy** and has the same `cp <source> <destination>` semantics as move, **except in the case of copying directories**. The reason for this is even though it is “free” (e.g. doesn’t take up extra disk space) to move directories, copying directories may involve a very large amount of

storage space, so you must confirm this action.

To copy directories, you need to use the `-r` flag, which stands for a **recursive copy**.

If you try to copy a directory in bash (Linux/macOS) without `-r`, you'll get an error:

```
$ ls
test_dir
$ ls test_dir
test1.txt
$ cp test_dir another_dir
cp: -r not specified; omitting directory 'test_dir'
```

If you try to copy a directory in Powershell (Windows), you won't get an error, but the newly "copied" directory will be empty.

In both cases, add the `-r` flag to copy directories:

```
$ ls
test_dir
$ ls test_dir
test1.txt
$ cp -r test_dir another_dir
$ ls
test_dir  another_dir
```

Our final command is what we use to delete files:

Command: **rm**

`rm` stands for **remove**, and it **permanently deletes files!** There is no recycle bin; things get immediately deleted. If you truly delete something important, then you'll need to try various data recovery techniques to get it back.

Much like `cp`, when deleting directories you need to add the `-r` (recursive) flag. If you don't, you'll either get an error message (Linux, macOS) or a confirmation dialog (Windows).

```
$ ls
test_dir  file1.txt  file2.txt
$ rm file*.txt
$ ls
test_dir
$ rm -r test_dir
```

Editing and viewing files

The last part of this covers basic file editing. You'll often be editing using some other GUI tool, but it's helpful to know how to use a basic command-line editor, especially when viewing files on computing clusters.

To print out the contents of a file to the terminal, you can use the `cat` command:

Command: `cat`

`cat` stands for **concatenate**. While it can still be used to combine multiple files together, it is more often used to print the contents of a file to the screen.

```
$ ls
text_file.txt
$ cat text_file.txt
This is the file's contents!
```

How do we actually edit files? There are many different command-line editors, but the most commonly installed ones are `vi` and `nano`, which typically both come preinstalled on Linux and MacOS. `nano` is much more user-friendly; `vi` and its predecessor `vim` are often much faster at day-to-day typing, but come with a steep (vertical?) learning curve.

Note: Windows comes without a command line text editor installed. If you have installed Git though, we can adjust your Powershell profile to have access to `nano`. Assuming Git has been installed to the default location you can do the following in Powershell:

```
> Set-Alias nano 'C:\Program Files\Git\usr\bin\nano.exe'
> nano $profile
  C:\Users\Username\Documents\WindowsPowerShell\Microsoft.PowerShell_
  <-profile.ps1

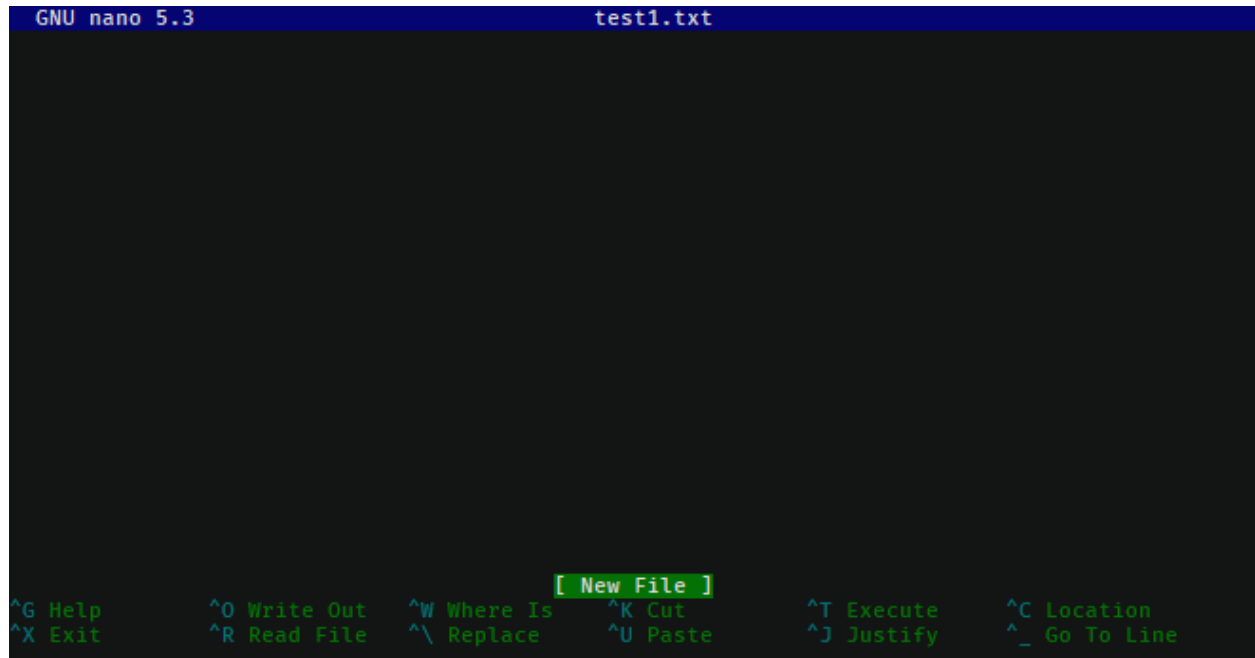
[ Read 1 lines (Converted from DOS format) ]
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C _
<-Location
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ _
<-Go To Line
```

Once in the editor, type the line `Set-Alias nano 'C:\Program Files\Git\usr\bin\nano.exe'`, then press `Ctrl-O` (denoted `^O` in the bottom menu), press enter, accepting the file-name, then press `Ctrl-X` to exit.

What this did was define what happens when you type `nano`; adding it to your profile script means that every Powershell session you start will have access to `nano`.

When opening Nano, you'll see the name of the file at the top. In the middle is your editing area; you can type and navigate around with arrow keys as you would expect.

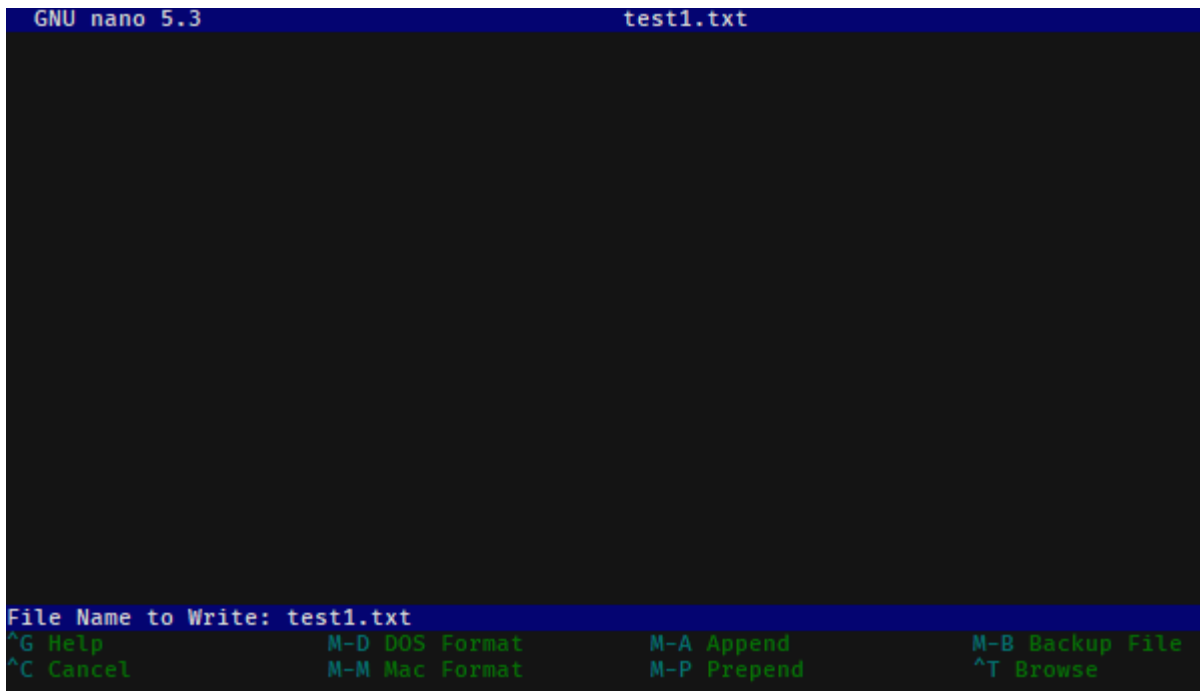
If you want to just open a blank editor, you can just type `nano`. If you want to edit a specific pre-existing file, then you can type `nano filename`.



At the bottom, you have a status bar that shows the available actions you can take. Here, control is represented with `^`, so `^X Exit` means you can press `Ctrl-X` to close Nano.

The important features are `Ctrl-X` to exit, `Ctrl-O` to save (“write out”), `Ctrl-W` for searching/find, and `Ctrl-\` for find-and-replace.

When you go to save a file after editing, you will bring up a bottom bar prompting you to give a filename. If you want to save with the same name as you opened, you can just hit enter; it auto-fills the current name. Edit the name if you want to save under a different filename.



```
GNU nano 5.3 test1.txt

File Name to Write: test1.txt
^G Help      M-D DOS Format  M-A Append     M-B Backup File
^C Cancel    M-M Mac Format  M-P Prepend     ^T Browse
```

Finding things in documents

`grep` is the tool you should use when searching through files. It lets you do basic searches and also regular-expression searches on one or multiple files.

Note: Windows also doesn't come with `grep` preinstalled. Add an alias to the Git-installed version as you did for `nano`:

```
> Set-Alias grep 'C:\Program Files\Git\usr\bin\grep.exe'
> nano $profile # Add the above line to your profile and save.
```

Command: `grep`

`grep`'s name comes from commands inside `ed`, one of the first text editors ever created, where it means 'global/regular expression/print'.

In short, it searches through entire files (global), using regular expressions as needed, and prints the search results it finds.

It's syntax is:

```
grep <flags> <match_pattern> <files>
```

When run without arguments, `grep` attempts to find `match_pattern` anywhere in the given files. Limited regular expression syntax is supported in this mode.

Following the example in the [Software Carpentries example](#)³⁷, consider searching through a file `haikus.txt` for the word `not`:

```
$ cat haikus.txt
The Tao that is seen
Is not the true Tao, until
You bring fresh toner.

With searching comes loss
and the presence of absence:
"My Thesis" not found.

Yesterday it worked
Today it is not working
Software is like that.
$ grep not haikus.txt
Is not the true Tao, until
"My Thesis" not found
Today it is not working
```

By default, `grep` returns any line that contains the given pattern.

If you want to use regex matching, you should pass the `-E` flag, but that is beyond the scope here.

Other useful options are `-w` for “word” matching and `-n` for line numbers.

Let’s consider what happens if we search for the word `is`:

```
$ grep is haikus.txt
The Tao that is seen
"My Thesis" not found.
Today it is not working
Software is like that.
```

It returned the second line because ‘is’ occurs inside the word `Thesis`! Adding the word flag forces `grep` to only match whole words:

```
$ grep -w is haikus.txt
The Tao that is seen
Today it is not working
Software is like that.
```

If we want to know more about where these lines were matched from, adding `-n` will give line numbers corresponding to lines returned:

```
$ grep -w -n is haikus.txt
1:The Tao that is seen
10:Today it is not working
11:Software is like that.
```

Finally, by adding the `-r`, the **recursive flag**, you can tell `grep` to search entire folders!

³⁷ <https://swcarpentry.github.io/shell-novice/07-find/index.html>

Exercise

Time for an exercise to test all of this out! Partner up with someone if desired.

After downloading this [zip file](#), unzip it **without looking inside the ``content`` directory!** Then, open up a terminal and get started! For some of the file manipulation tasks, there is a helper Python script that can check if you have correctly completed parts of the exercise. Run it with `python check.py`.

Exercise

Do all of the following just in your terminal, without using GUI tools.

1. What directory did your terminal start in?
2. Where did you unzip the exercise directory to?

Use `cd` and `ls` to navigate into the `content` directory.

3. What is the directory layout of the `content` directory? You may want to draw out the directory tree.
4. Are there any hidden folders?
5. In the `data` directory, the `imaging` subdirectory was accidentally placed inside the `sequencing` directory; move it out so it is next to the `sequencing` directory.
6. Add your name to the `README.txt` file.
7. In the `data/mixed` directory, there is a mixture of `.fastq` and `.tif`, and `.fcs` files. Create a new directory, `flow`, and **move** these files into the `sequencing`, `imaging`, and `flow` directories, respectively. Delete the `data/mixed` directory after everything has been moved out.
8. In the `raw` directory, there are mixed data files from several years. Create a `raw` directory inside `data`, then create `YYYY` directories for each year (e.g. 2020), sorting the mixed data files by **copying** them into the year directories.
9. In the `primers` directory, there are (hint: use `grep`!). Which primers contain the repetitive sequence `AAAA`?
 1. Will change per user.
 2. Will also depend per user, but probably in their Download folder.
 3. The directory layout is as follows:

```
content
├── data
│   ├── mixed
│   ├── sequencing
│   │   └── imaging
├── primers
├── raw
│   └── 2019.08.06_SlowFT_reprogram
```

(continues on next page)

(continued from previous page)

```
├── 2020-02-19-PEI-titration
├── scripts
│   ├── .git
│   └── env
```

4. There are hidden folders, the `scripts/.git` folder.
5. From inside content, this can be done with `mv data/sequencing/imaging data/`, or other ways.
6. This can be done with `nano README.txt`.
7. This can be done by using `mv` with a wildcard file specification. One solution is:

```
$ cd data
$ mkdir flow
$ mv mixed/*.fcs flow
$ mv mixed/*.tif imaging
$ mv mixed/*.fastq sequencing
$ rm mixed
```

8. This can again be done with a combination of `mkdir` and `mv`, with file specifications like `2020*`.

Starting from the content folder:

```
$ cd raw
$ mkdir ../data/raw
$ mkdir ../data/raw/2019
$ mkdir ../data/raw/2020
$ mkdir ../data/raw/2021
$ cp -r 2019* ../data/raw/2019
$ cp -r 2020* ../data/raw/2020
$ cp -r 2021* ../data/raw/2021
```

9. The following primers have quadruple A's in a row:

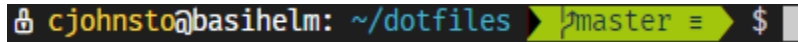
```
geec_primers.txt:oGE024 Rsites-UbC_fwd agtccagtgtCATCAACAAGTTTGTACAAAAAAG
nbw_primers.txt:GG_lenti_bb_fwd ACAGCGTCTCAtcctTAAAAGAAAAGGGGGGAC
nbw_primers.txt:GG_WPRE_fwd ACAGCGTCTCAttgcCGATAATCAACCTCTGGATTACAAAATT
nbw_primers.txt:link-NLS-VPR_rev
→gaaagctgggtctagatatcTCAAACAGAGATGTGTCTGAAGATGG
nbw_primers.txt:"mCherry-MCP-VP16_rev "
→gaaagctgggtctagatatcCTACAGCATATCCAGATCAAAATCGTC
nbw_primers.txt:Rsites-UbC_MCS_fwd
→tccagtttgggcatgcgctagcctcgagGCATCAACAAGTTTGTACAAAAAAG
nbw_primers.txt:VPR_rev gaaagctgggtctagatatcTCAAACAGAGATGTGTCT
sequencing_primers.txt:oSEQ046 phage-dest-upstream-seq CGACGTACTCCAAAAGCTCGAG
```

What directory did you unzip into?

Extras

If you'd like to customize your shell prompt so it is more useful, you can use something called [Oh my Posh 3](#)⁴¹.

For example, my terminal looks like this when logged into a remote server:



First, it shows a lock symbol because I'm connected over a secure connection, followed by my username and the server name. In blue, the current working directory is shown. Finally, it shows git status directly on the prompt line (here, we are on branch `master` with no changes).

Before installing this, make sure you have a powerline-enabled font (like Fira Code). Then, follow the installation instructions [here](#)⁴². If you are on Windows, this is simple, just type:

```
> Install-Module oh-my-posh -Scope CurrentUser -AllowPrerelease
```

Then, create a JSON file with your desired prompt; you can do this with `nano ~/.omp_prompt.json`.

After this, there is typically a [final setup step](#)⁴³ that modifies your profile file and points it at your prompt file. On Powershell, this is `Set-PoshPrompt ~/.omp_prompt.json`, added via `nano $profile`. On Linux and MacOS, this is typically adding

```
eval "$(oh-my-posh --init --shell zsh --config ~/.poshthemes/jandedobbeleer.omp.json)"
```

to either your `.zshrc` (`nano ~/.zshrc`) or `.bashrc` (`nano ~/.bashrc`), depending on which shell you use.

If you'd like to copy my prompt, mine is:

```
{
  "final_space": false,
  "blocks": [
    {
      "type": "prompt",
      "alignment": "left",
      "segments": [
        {
          "type": "session",
          "style": "plain",
          "foreground": "#ffffff",
          "properties": {
            "postfix": ":",
            "user_color": "#ffd93d",
            "ssh_icon": "\uE0A2 "
          }
        },
        {
          "type": "path",
```

(continues on next page)

⁴¹ <https://ohmyposh.dev/>

⁴² <https://ohmyposh.dev/docs/installation>

⁴³ <https://ohmyposh.dev/docs/installation/#4-replace-your-existing-prompt>

(continued from previous page)

```
"style": "plain",
"foreground": "#44B4CC",
"properties": {
  "style": "agnoster_full"
},
{
  "type": "git",
  "style": "powerline",
  "powerline_symbol": "\uE0B0",
  "foreground": "#193549",
  "background": "#alc60b",
  "properties": {
    "local_working_icon": " W",
    "local_staged_icon": " S"
  }
},
{
  "type": "python",
  "style": "powerline",
  "background": "#00897b",
  "foreground": "#193549",
  "powerline_symbol": "\uE0B0",
  "properties": {
    "display_version": false,
    "display_virtual_env": true,
    "display_mode": "context"
  }
},
{
  "type": "text",
  "style": "plain",
  "properties": {
    "text": "$"
  }
}
]
}
```

TODO

Add IAP bootcamp info

CONTRIBUTOR GUIDE

Protocols and recipes are written in reStructuredText, a lightweight markup language that allows us to use plain text to describe relatively complicated intent, letting us make tables, cross-links between different files, image inclusion, and so on.

The Python package Sphinx is used to automatically take this information and make a [searchable website](#)⁴⁴, in addition to a printable PDF version of all of the protocols and recipes, available [here](#)⁴⁵.

If you want to make small edits or prefer the browser interface, you can skip all local setup and go directly to [Online editing through Github](#) (page 93).

If you already are familiar with text editors and have an editing/programming environment already setup, you can go directly to [Local building](#) (page 93).

5.1 Local environment setup

To locally build the protocols documentation, you will need to have a text editor of your choice and a local install of Python. To contribute your changes, you will need access to **git** as well.

5.1.1 Text editor

Unlike a what-you-see-is-what-you-get (WYSIWYG) editor like Word, a (*plain*) text editor is used for editing markup and programming languages.

Note: Integrated development environments (IDEs) such as MATLAB, RStudio, TeXStudio, and others typically have a text editor integrated with debugging and other tools. These are helpful, but are often single-purpose. Having a text editor customized to your liking is especially helpful when switching between many languages or if extra customization is desired.

There are many options available:

⁴⁴ <https://gallowaylabmit.github.io/protocols>

⁴⁵ https://gallowaylabmit.github.io/protocols/galloway_lab_protocols.pdf

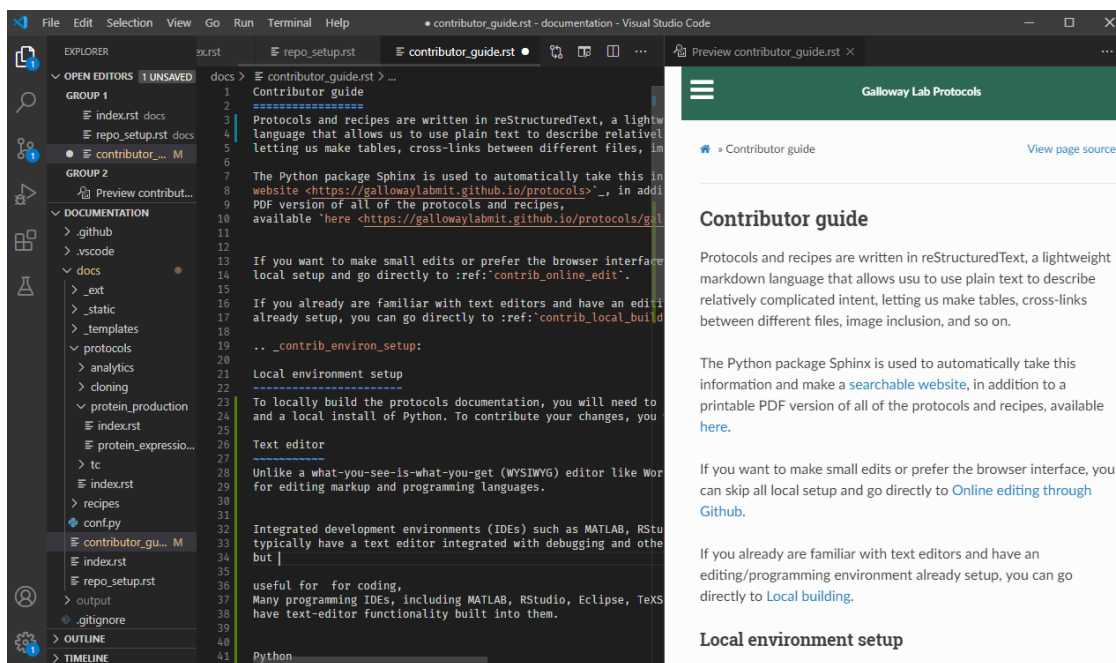
- **Built-in editors:** Every major OS comes pre-installed with a plain text editor. Windows has *notepad*, Mac OSX has *TextEdit* (while this defaults to a rich-text editor, it can also be used as a plain-text editor), and most Linux distros have *vi* or *GEdit*.
- **Terminal-based editors:** *nano* is a basic editor, whereas *vim* and *emacs* are extraordinarily customizable.
- **Standalone editors:** These editors can be customized to be nearly IDE-like. On Windows, there is the venerable *Notepad++*⁴⁶. More recently, there are the commonly used editors *Sublime Text*⁴⁷, *Atom*⁴⁸, and *VS Code*⁴⁹.

Choice of editor is largely a personal preference. If you are doing limited plain-text editing, the built-in editors may be sufficient. Becoming familiar with one of the terminal-based editors is useful when working on remote compute clusters and servers.

If you are looking for a recommendation, VS Code is an excellent, relatively lightweight text editor with plenty of helpful extensions.

Tip: Using a modern text editor is very helpful when working with markup languages such as reStructuredText, Markdown, and LaTeX, as they often have support for live document preview, intelligent spell-check (ignoring programming terms as mis-spelled), and git support (no need to use the git command line or GUI app).

As an example, this document was written in VSCode, with the Python and reStructuredText extensions installed:



⁴⁶ <https://notepad-plus-plus.org/>

⁴⁷ <https://www.sublimetext.com/>

⁴⁸ <https://atom.io/>

⁴⁹ <https://code.visualstudio.com/>

5.1.2 Python/Sphinx

[Sphinx](https://www.sphinx-doc.org/en/master/)⁵⁰ is used to create the rendered website and PDF. Sphinx relies on a Python version at least as new as Python 3.5.

If you do not already have a working Python version ≥ 3.5 , use the [standard Python installer](https://www.python.org/)⁵¹.

Then, install Sphinx and the required Sphinx extensions used here using `pip` (enter the following **without** the leading `$`):

```
$ pip install -U sphinx sphinx-rtd-theme sphinx-last-updated-by-git
```

⁵⁰ <https://www.sphinx-doc.org/en/master/>

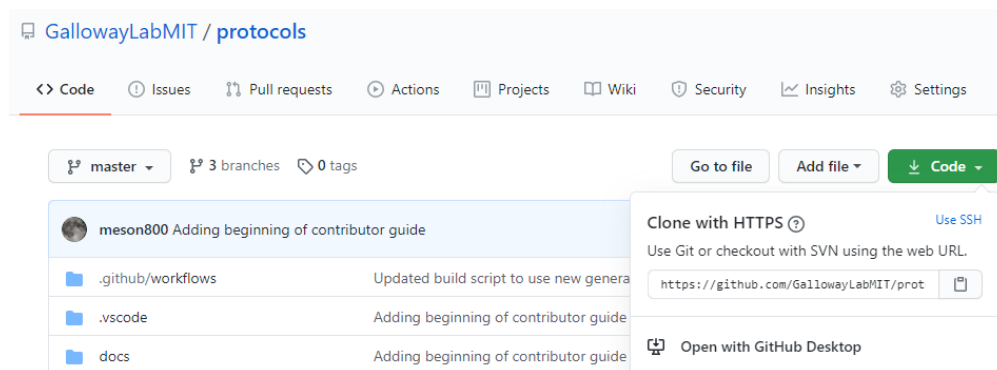
⁵¹ <https://www.python.org/>

5.1.3 git

We use `git` to manage version history, simultaneous editing, and other features. There are [several](#)⁵² [excellent](#)⁵³ [tutorials](#)⁵⁴ elsewhere that explain how to use `git`.

If you don't have `git`, you can [install it from here](#)⁵⁵, and install a GUI tool if you wish (such as the standalone [Github Desktop](#)⁵⁶, or using one built-in to your text editor).

Once you have `git` installed, you should *clone* the protocols repository. For any Github repository, you can find the clone URL by clicking the green “code” button:



In the case of this repository, the HTTPS clone URL is <https://github.com/GallowayLabMIT/protocols.git>.

If you access Github using `ssh` keys, the SSH clone URL is [git@github.com:GallowayLabMIT/protocols.git](ssh://git@github.com:GallowayLabMIT/protocols.git).

The published version of the website uses the default `master` branch, so push to this branch to update the website. A normal workflow to update a protocol would be:

1. Make changes to the desired files, such as adding pictures, writing new text, and so on.
2. Locally build the protocols website, checking for any errors (e.g. incorrect reStructuredText)
3. When there are no build errors, add the files and create a commit describing your changes.
4. Do a `git pull` to merge any new changes, followed by a `git push` to update the website.

⁵² <https://git-scm.com/book/en/v2>

⁵³ <https://try.github.io/>

⁵⁴ <https://gitimmersion.com/>

⁵⁵ <https://git-scm.com/downloads>

⁵⁶ <https://desktop.github.com/>

5.2 Local building

The documentation can be built by calling the `build.py` script at the base of this repository. Normally, this means opening a terminal window, navigating to the repository, and calling:

```
python build.py
```

This will attempt to build both the website and the PDF. If you do not have a local LaTeX install (e.g. you either do not use LaTeX, or you exclusively use online services like Overleaf), then local PDF building will fail. If this is the case, you can skip PDF builds by calling

```
python build.py --skip-latex
```

These build functions build the website in the folder `output/html`. If you want to view your locally built website, open the file `output/html/index.html`.

In the case of strange build errors that seem to be because the output directory has been corrupted, you can close any program that might be using the output (a common one might be Adobe Acrobat, with the generated PDF open) and run:

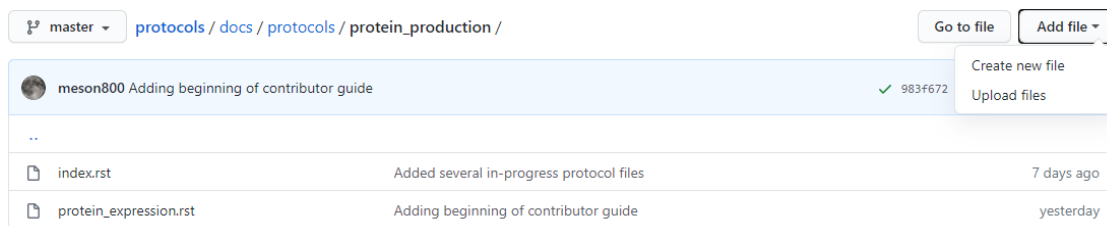
```
python build.py --force-rebuild
```

Adding this flag deletes the `output` folder and recreates it. You can also do this manually to recreate “rebuild” behavior.

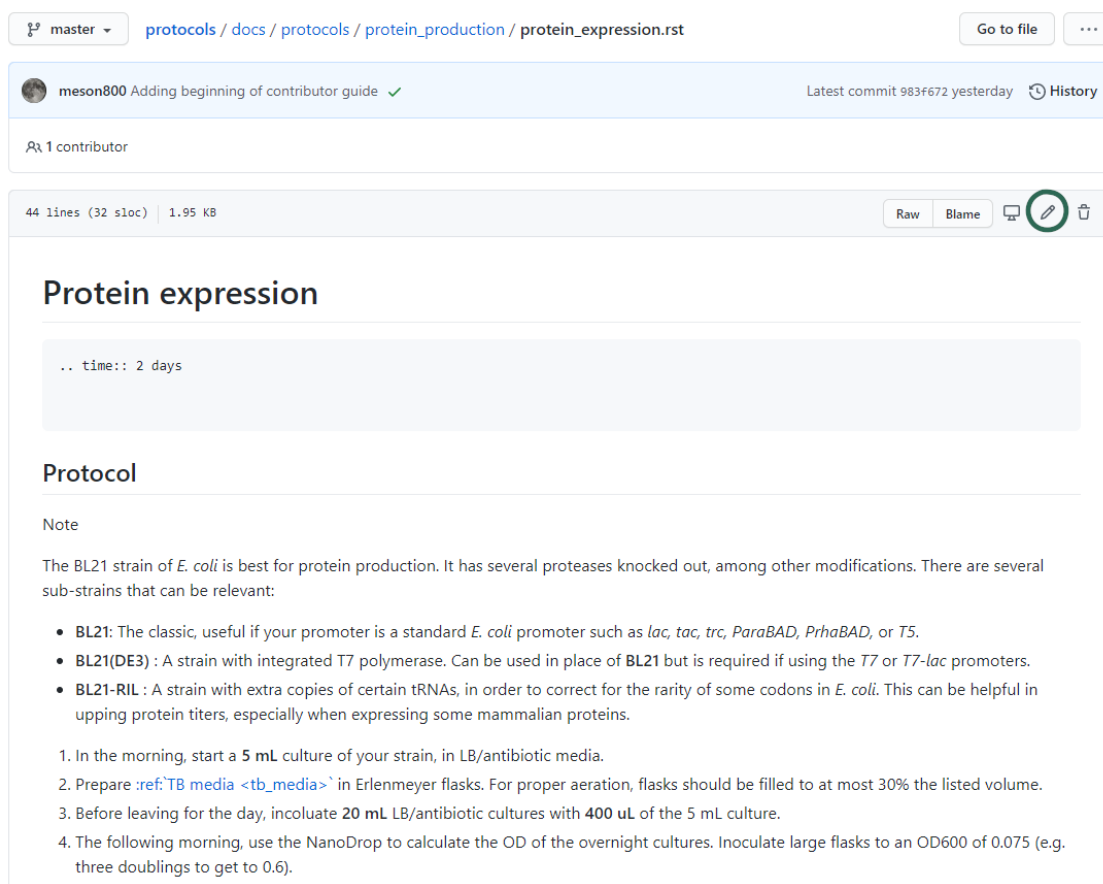
5.3 Online editing through Github

When editing directly through the Github website, you won’t be able to check for Sphinx build errors or fully preview the generated PDF and website until you commit to the branch. For this reason, doing *local builds* (page 93) is preferred.

To create a new file directly through Github, navigate to the folder you want to add the file, and click the **Add file** dropdown on the right:



To edit through Github, navigate to the file you want to edit, then click the pencil in the upper right of the file view:



Note: While Github does render a preview of what the reStructuredText will look like, it does not preview how Sphinx will render the final website. For example, we can see in the above image that the Github preview does not show the custom `time` directive, and it does not show the proper link destination for the cross-referenced recipe.

In general, the Github preview will give you a good idea of what tables/lists/other text will appear, but it will not properly render all Sphinx-enabled markup.

This will open an editor window:

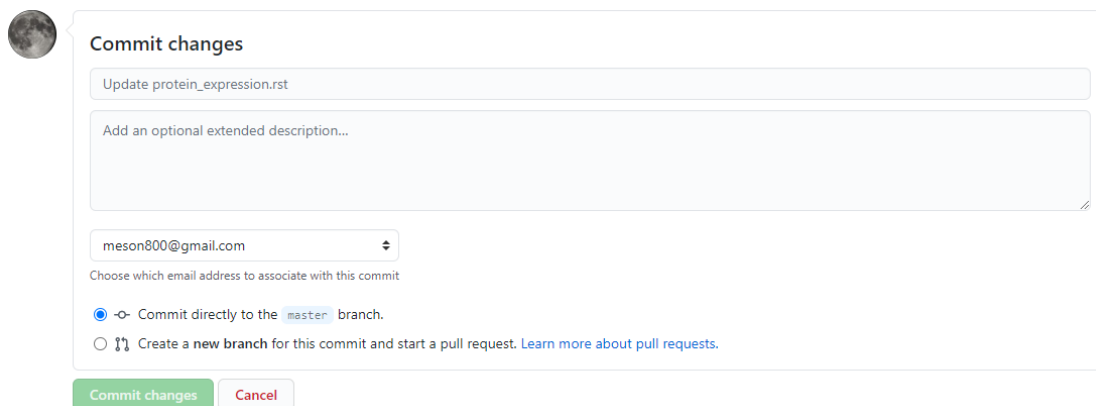
protocols / docs / protocols / protein_production / protein_expression.rst Cancel

```

1 |=====
2 Protein expression
3 |=====
4
5 .. time:: 2 days
6
7
8 Protocol
9 =====
10 .. note::
11     The BL21 strain of E. coli* is best for protein production. It has several proteases knocked
12     out, among other modifications. There are several sub-strains that can be relevant:
13
14     - BL21*: The classic, useful if your promoter is a standard E. coli* promoter
15       such as lac, tac, trc, ParaBAD, PrhaBAD,* or T5*.
16     - BL21(DE3)*: A strain with integrated T7 polymerase. Can be used in place of BL21*
17       but is required if using the T7* or T7-Lac* promoters.
18     - BL21-RIL*: A strain with extra copies of certain tRNAs, in order to correct for the
19       rarity of some codons in E. coli*. This can be helpful in upping protein titers,
20       especially when expressing some mammalian proteins.
21
22 1. In the morning, start a 5 mL* culture of your strain, in LB/antibiotic media.
23 2. Prepare ref: TB media <tb_media> in Erlenmeyer flasks. For proper aeration,
24   flasks should be filled to at most 30% the listed volume.
25 3. Before leaving for the day, inoculate 20 mL* LB/antibiotic cultures with 400 uL* of the 5 mL culture.
26 4. The following morning, use the NanoDrop to calculate the OD of the overnight cultures. Inoculate large
27   flasks to an OD600 of 0.075 (e.g. three doublings to get to 0.6).

```

After you are done editing, add a commit message describing your change, and (normally), commit directly to the `master` branch. If there is need for further discussion of an added protocol, creating a secondary branch + pull request could be helpful.



Commit changes

Update protein_expression.rst

Add an optional extended description...

meson800@gmail.com

Choose which email address to associate with this commit

☒ Commit directly to the `master` branch.

☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

Commit changes Cancel

Note: Make sure you make the commit message more descriptive than the default “Update <file-name>” message!

5.4 Repository layout

All of the relevant documentation files to edit are stored in the `docs` folder.

Each subdirectory is included as its own sub-level in the table of contents. This hierarchy is derived from the `index.rst` that is in each folder. Generally, each of these `index.rst` files have the following content; if you create a new subdirectory you should generally add this as the `index.rst` file:

```
=====  
Section name  
=====
```

```
.. toctree::  
    :maxdepth: 2  
    :glob:
```

```
*/index  
*
```

Here, `glob` means that the `*` is expanded as a wildcard. The first wildcard search, `*/index` means “include all subdirectories beneath this directory”. The second wildcard search `*` means “include all other `.rst` files in this directory”.

The current subdirectory layout looks like:

```
docs  
├── protocols  
│   ├── analytics  
│   ├── cloning  
│   ├── protein_production  
│   └── tc  
├── recipes  
│   ├── bacteria  
│   └── tc
```

The `.github` folder contains the continuous integration script responsible for updating the website on every push.

5.5 Basics of reStructuredText

reStructuredText (RST) is a *lightweight* markup language. This means that it is not as cumbersome as languages like HTML and LaTeX, but still has enough power to make nice looking documents.

There is an [excellent RST primer and reference here](https://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html)⁵⁷ which should be your primary reference, but here we will cover some of the basics.

One nice feature of the generated website is the ability to view the page source for each page. If there is a protocol that uses some RST feature that you want to replicate, on the desktop version of the website (e.g. not mobile), you can click the “View page source” button in the upper right of **any page** on the website to see the RST code that generated that page. That includes this page, so check out this page’s source to see how this guide was written!

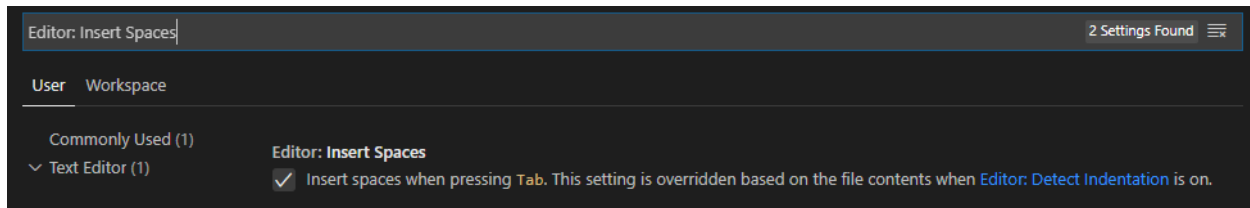
Note: There are several whitespace-dependent features of RST. This means that you should configure your text editor to insert spaces instead of tabs when you hit the tab button (this is also true if you are programming in whitespace-dependent languages like Python).

⁵⁷ <https://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html>

Without wading too deeply into the [holy war](#)⁵⁸, tabs vs spaces **does not mean the difference between pressing the space bar vs hitting tab for indentation**, it refers to what character actually gets inserted into the document when you press the tab button.

Long story short, if your text editor inserts literal tab characters, there is possible inconsistency between tools and editors; some may display a single tab character as the width of two spaces, some as the width of four spaces, and so on. This causes problems. If you set your editor to insert spaces, you still hit tab, but the editor inserts some fixed number of spaces, typically four.

This setting will depend per editor. In VS code for example, you don't have to do anything; it defaults to inserting spaces, but the option looks like this:



5.5.1 Simple markup

You can add headers by surrounding the header with equal signs, hyphens, tildes, and other special characters.

Example

```
Section header
=====

Subsection header
-----
```

Use single asterisks to italicize text. Use double asterisks to bold text. Wrapping double backticks around text renders it in monospace.

Example

```
*italics* and **bold** and ``monospaced``.
```

renders as

italics and **bold** and monospaced.

You can make bullet lists by starting lines with `*` or `#`, and you can make numbered lists by starting lines with `1.`, `2.`, etc.

⁵⁸ <https://softwareengineering.stackexchange.com/questions/57/tabs-versus-spaces-what-is-the-proper-indentation-character-for-ever>

If you are nesting lists, you must surround nesting levels with blank lines.

Example

```
* Lab activities

  * Axe throwing
  * Pizza party
  * ?????

* Lab meme sources

  * p53
  * Cloning, so much cloning

1. Testing
2. a
3. numbered
4. list
```

renders as

- Lab activities
 - Axe throwing
 - Pizza party
 - Other?
 - Lab meme sources
 - p53
 - Cloning, so much cloning
1. Testing
 2. a
 3. numbered
 4. list
-

5.5.2 Explicit markup

In RST, an “explicit” block is any block that starts with `...`. Explicit blocks must be surrounded on both sides by blank lines, like nested lists. This means that explicit blocks like:

```
Do the foo, then the bar
.. note::

    Make sure you don't do the bar, then the foo!
```

will not render correctly, it must be written:

```
Do the foo, then the bar

.. note::

    Make sure you don't do the bar, then the foo!
```

5.5.3 Admonitions

To call-out a specific part of a protocol, you can use one of the various admonitions.

This project includes the special time estimation admonition, which demonstrates the general principle. Writing:

```
.. time::  
  
    2 hours
```

renders as

Estimated time: 2 hours

Other directives (code blocks, tables, images, etc) can be fully nested inside these blocks.

Other options commonly used here are `hint`, `important`, `note`, `tip`, and `warning`, which render as

Hint: Test

Important: Test

Note: Test

Tip: Test

Warning: Test

5.5.4 Code

To insert a codeblock, start an empty line with `::`, followed by an indented block that will be rendered as code.

Example

The above code example of the time estimation admonition was written as:

```
::  
  
.. time::  
  
    2 hours
```

5.5.5 Tables

See the [table documentation](#)⁵⁹ for more details, but in brief, most of the time you can use the “simple” table layout.

In the simple table layout, you simply surround the desired table text with equal signs to set off different columns.

Example

```
=====
Ingredient      Amount per 1L final volume
=====
Tryptone        12 g
Yeast extract   24 g
Glycerol        4 mL
Deionized water  900 mL
=====
```

renders as

Ingredient	Amount per 1L final volume
Tryptone	12 g
Yeast extract	24 g
Glycerol	4 mL
Deionized water	900 mL

⁵⁹ <https://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html#tables>

5.5.6 References and links

To reference a standalone hyperlink, you can just simply write it directly in the rst file, no special markup required.

If you want to add link text, use the following syntax:

Example

```
https://example.org or `Link text <https://example.org>`_
```

renders as:

<https://example.org> or [Link text](https://example.org)⁶⁰

If you want to specifically link to other protocols/recipes files, you use the special *doc* syntax:

Example

```
:doc:`This <contributor_guide>` is a link to this very document!
```

renders as:

[This](#) (page 89) is a link to this very document!

as you can see, this is very similar to the external hyperlink, except it has the special `:doc:` before it, which tells Sphinx that the document is included in this repository.

If you want to reference a specific subsection of a document, you can set a label and set a reference to it. This is best explained by the [documentation](#)⁶¹, noting that labels that you create are global and shared across the entire repository!

⁶⁰ <https://example.org>

⁶¹ <https://www.sphinx-doc.org/en/master/usage/restructuredtext/roles.html#ref-role>

5.5.7 Images

To include an image, you can specify it as follows. Typically, you want to align center and make the image fill the available horizontal space, so a common image call would be:

```
.. image:: image_location/image_filename.png
   :width: 100%
   :align: center
```

5.5.8 Math

You can write arbitrary LaTeX-formatted math by using the `math` directive. The math must be separated from the directive by a blank line, followed by an indented math block.

Example

```
.. math::  
  
    E = mc^2
```

renders as

$$E = mc^2$$

CONTRIBUTING

See the [Repository setup guide](#) (page 47) if you are interested in how the continuous build system works, or you'd like to fork this repository to make it your own.

See the [Contributor guide](#) (page 89) for a more detailed walkthrough of how to contribute to this protocols repository.