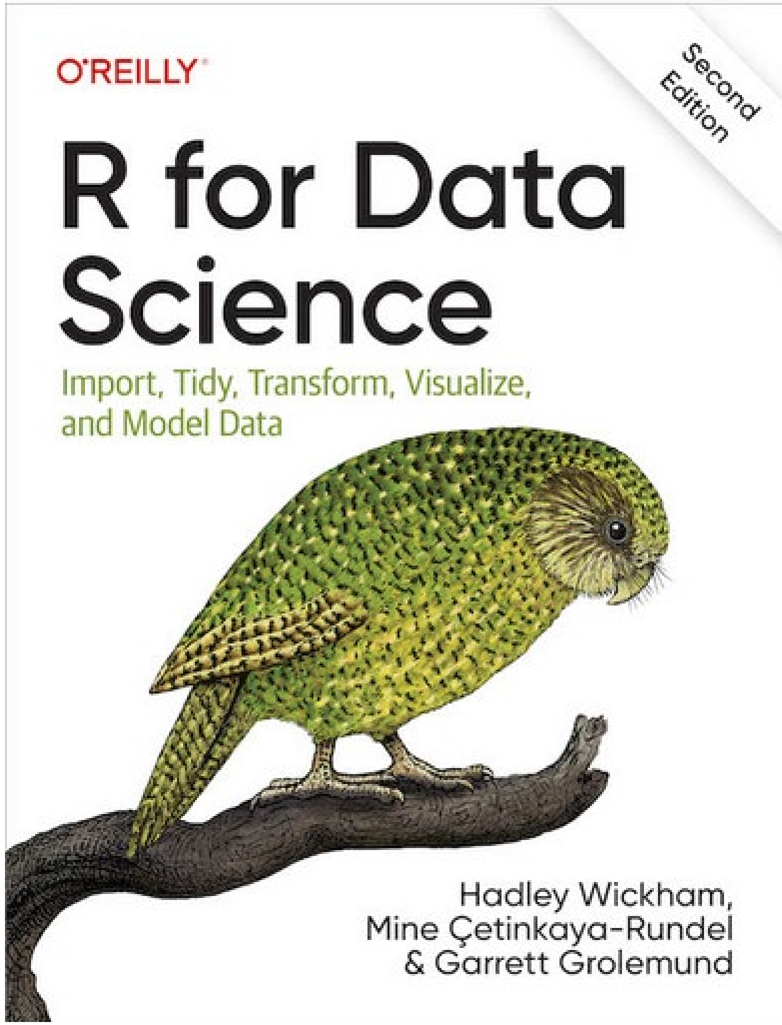
Data Tidying and Visualization

Justin Savage

11/6/24

Great Data Science Resources



O'REILLY

Second Edition

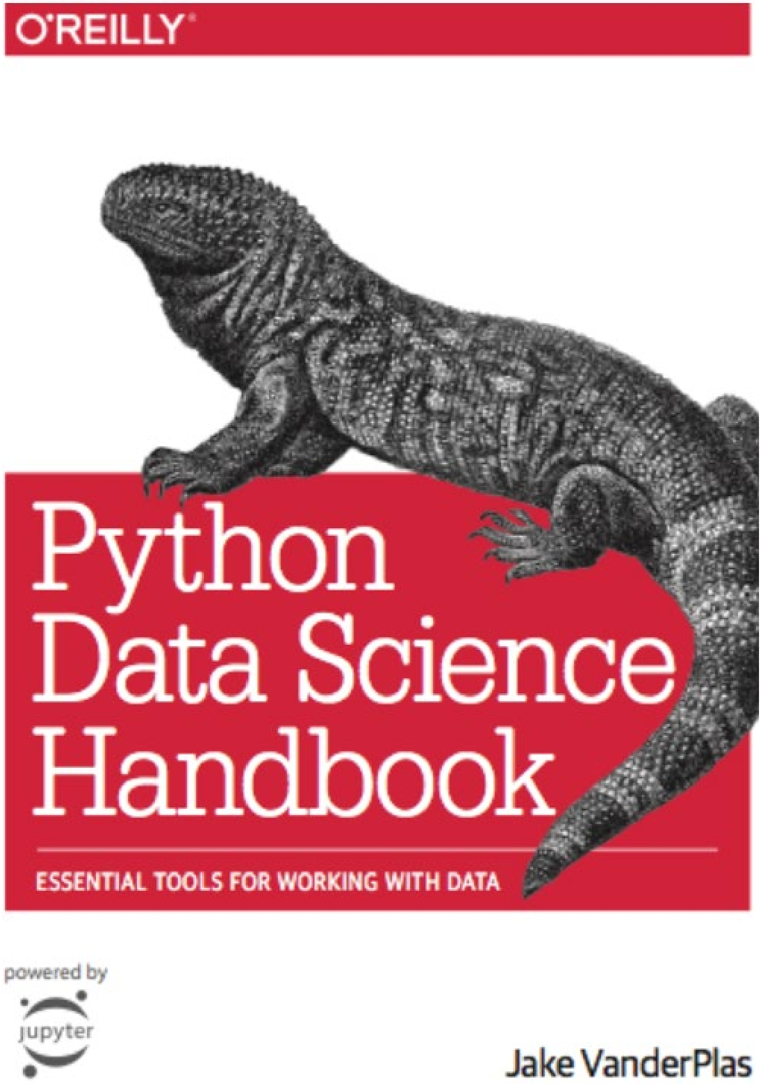
R for Data Science

Import, Tidy, Transform, Visualize, and Model Data

Hadley Wickham,
Mine Çetinkaya-Rundel
& Garrett Grolemund

https://r4ds.hadley.nz/

https://jakevdp.github.io/PythonDataScienceHandbook/



O'REILLY

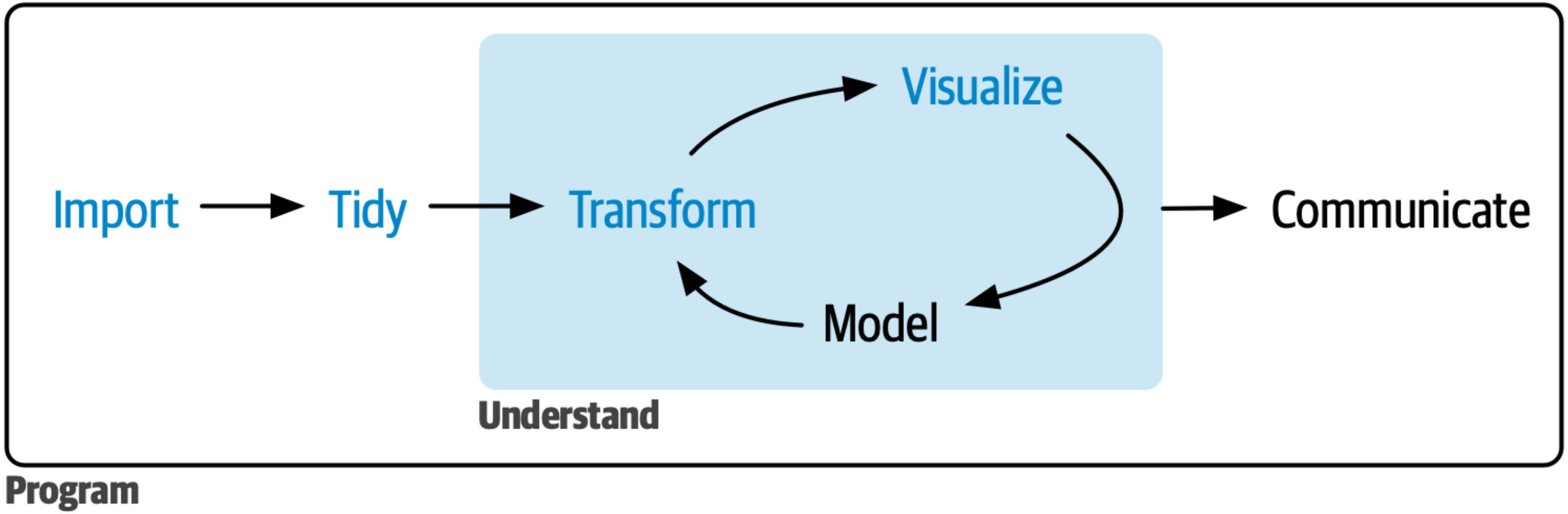
Python Data Science Handbook

ESSENTIAL TOOLS FOR WORKING WITH DATA

powered by jupyter

Jake VanderPlas

Data Analysis Workflow



Tabular Data

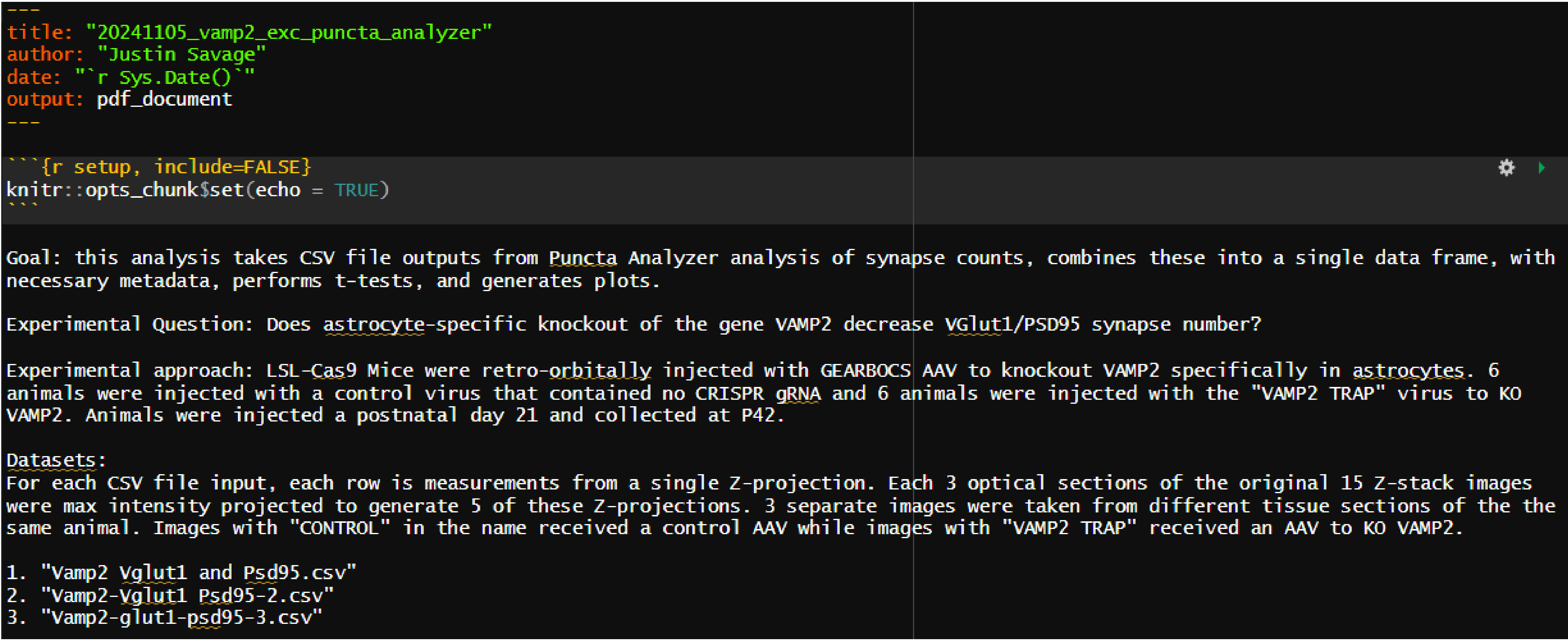
- Data in the form of a table or spreadsheet
- Includes:
    - Values – specific measurements for a given variable
    - Observations – a set of values that are related (i.e. from the same image)

- Tabular data is commonly stored in R as a DataFrame object

Loading Tabular Data

- Before loading data, it is good practice to set up an R Markdown file and write out the experimental details

Loading Tabular Data



~~~
---
title: "20241105_vamp2_exc_puncta_analyzer"
author: "Justin Savage"
date: "`r Sys.Date()`"
output: pdf_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
~~~

Goal: this analysis takes CSV file outputs from Puncta Analyzer analysis of synapse counts, combines these into a single data frame, with
necessary metadata, performs t-tests, and generates plots.

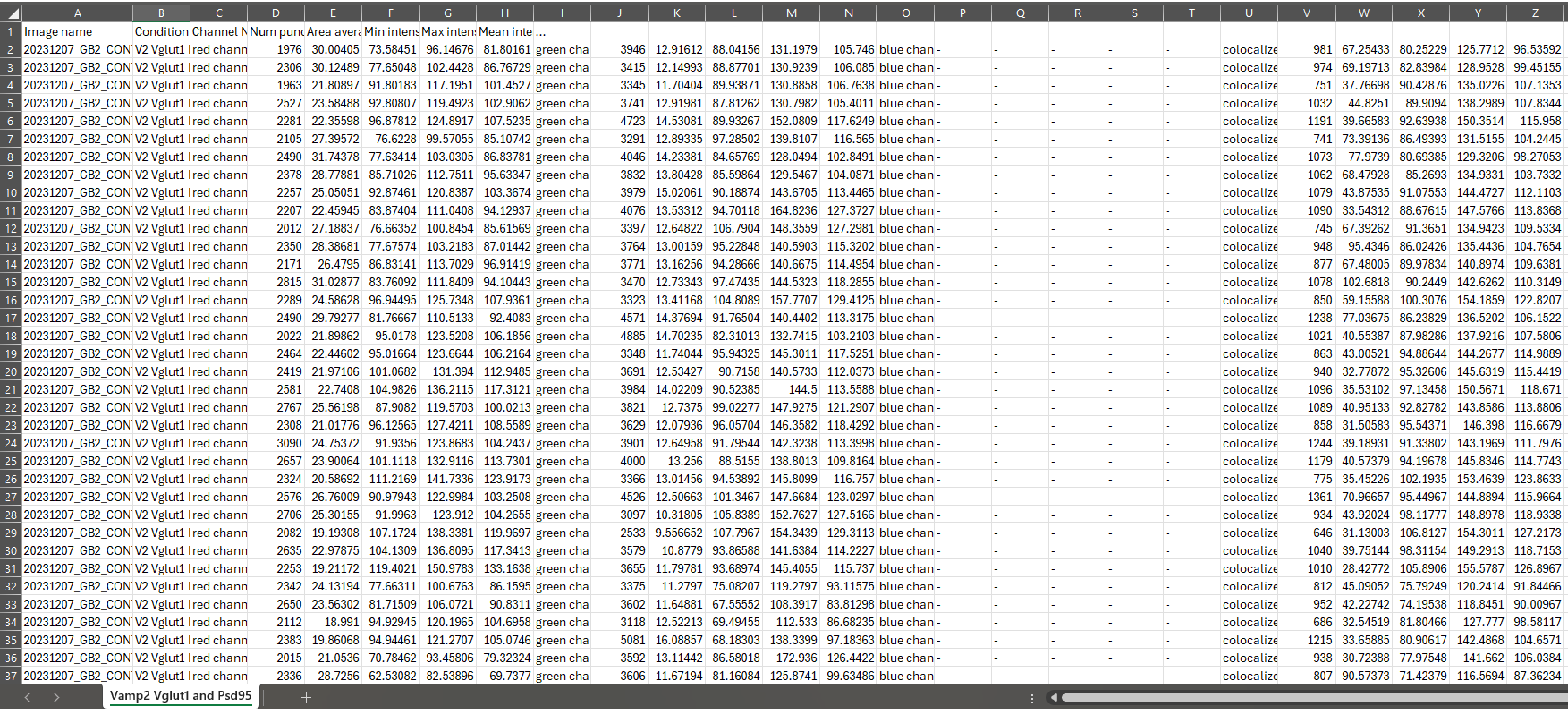
Experimental Question: Does astrocyte-specific knockout of the gene VAMP2 decrease VGlut1/PSD95 synapse number?

Experimental approach: LSL-Cas9 Mice were retro-orbitally injected with GEARBOCS AAV to knockout VAMP2 specifically in astrocytes. 6
animals were injected with a control virus that contained no CRISPR gRNA and 6 animals were injected with the "VAMP2 TRAP" virus to KO
VAMP2. Animals were injected a postnatal day 21 and collected at P42.

Datasets:
For each CSV file input, each row is measurements from a single Z-projection. Each 3 optical sections of the original 15 Z-stack images
were max intensity projected to generate 5 of these Z-projections. 3 separate images were taken from different tissue sections of the the
same animal. Images with "CONTROL" in the name received a control AAV while images with "VAMP2 TRAP" received an AAV to KO VAMP2.

1. "Vamp2 Vglut1 and Psd95.csv"
2. "Vamp2-Vglut1 Psd95-2.csv"
3. "Vamp2-glut1-psd95-3.csv"
```
```

Loading Tabular Data

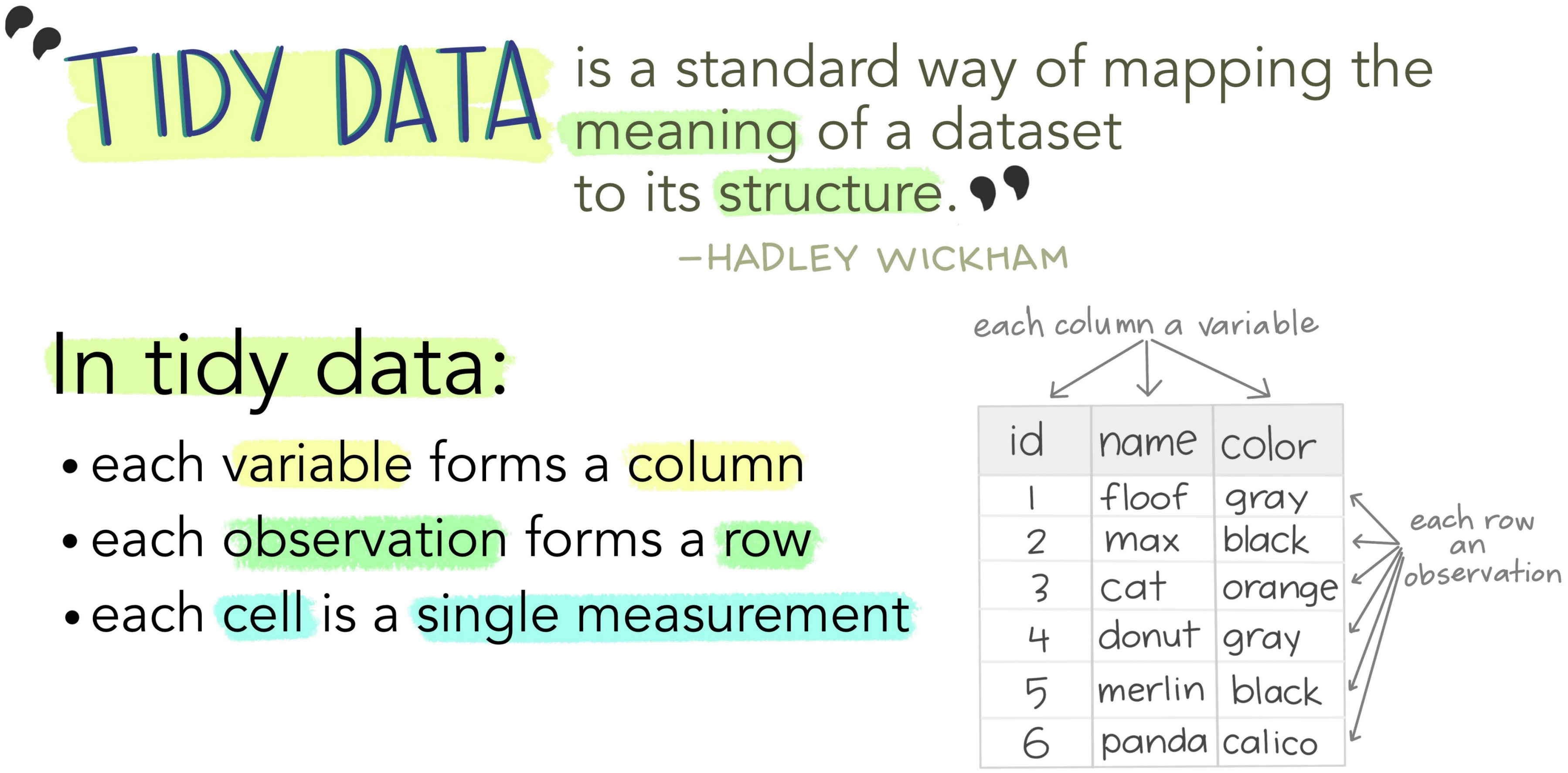


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	Image name	Condition	Channel N	Num punc	Area avera	Min intens	Max inten	Mean inte	...																	
2	20231207_GB2_CON	V2 Vglut1	red chann	1976	30.00405	73.58451	96.14676	81.80161	green cha	3946	12.91612	88.04156	131.1979	105.746	blue chan	-	-	-	-	-	colocalize	981	67.25433	80.25229	125.7712	96.53592
3	20231207_GB2_CON	V2 Vglut1	red chann	2306	30.12489	77.65048	102.4428	86.76729	green cha	3415	12.14993	88.87701	130.9239	106.085	blue chan	-	-	-	-	-	colocalize	974	69.19713	82.83984	128.9528	99.45155
4	20231207_GB2_CON	V2 Vglut1	red chann	1963	21.80897	91.80183	117.1951	101.4527	green cha	3345	11.70404	89.93871	130.8858	106.7638	blue chan	-	-	-	-	-	colocalize	751	37.76698	90.42876	135.0226	107.1353
5	20231207_GB2_CON	V2 Vglut1	red chann	2527	23.58488	92.80807	119.4923	102.9062	green cha	3741	12.91981	87.81262	130.7982	105.4011	blue chan	-	-	-	-	-	colocalize	1032	44.8251	89.9094	138.2989	107.8344
6	20231207_GB2_CON	V2 Vglut1	red chann	2281	22.35598	96.87812	124.8917	107.5235	green cha	4723	14.53081	89.93267	152.0809	117.6249	blue chan	-	-	-	-	-	colocalize	1191	39.66583	92.63938	150.3514	115.958
7	20231207_GB2_CON	V2 Vglut1	red chann	2105	27.39572	76.6228	99.57055	85.10742	green cha	3291	12.89335	97.28502	139.8107	116.565	blue chan	-	-	-	-	-	colocalize	741	73.39136	86.49393	131.5155	104.2445
8	20231207_GB2_CON	V2 Vglut1	red chann	2490	31.74378	77.63414	103.0305	86.83781	green cha	4046	14.23381	84.65769	128.0494	102.8491	blue chan	-	-	-	-	-	colocalize	1073	77.9739	80.69385	129.3206	98.27053
9	20231207_GB2_CON	V2 Vglut1	red chann	2378	28.77881	85.71026	112.7511	95.63347	green cha	3832	13.80428	85.59864	129.5467	104.0871	blue chan	-	-	-	-	-	colocalize	1062	68.47928	85.2693	134.9331	103.7332
10	20231207_GB2_CON	V2 Vglut1	red chann	2257	25.05051	92.87461	120.8387	103.3674	green cha	3979	15.02061	90.18874	143.6705	113.4465	blue chan	-	-	-	-	-	colocalize	1079	43.87535	91.07553	144.4727	112.1103
11	20231207_GB2_CON	V2 Vglut1	red chann	2207	22.45945	83.87404	111.0408	94.12937	green cha	4076	13.53312	94.70118	164.8236	127.3727	blue chan	-	-	-	-	-	colocalize	1090	33.54312	88.67615	147.5766	113.8368
12	20231207_GB2_CON	V2 Vglut1	red chann	2012	27.18837	76.66352	100.8454	85.61569	green cha	3397	12.64822	106.7904	148.3559	127.2981	blue chan	-	-	-	-	-	colocalize	745	67.39262	91.3651	134.9423	109.5334
13	20231207_GB2_CON	V2 Vglut1	red chann	2350	28.38681	77.67574	103.2183	87.01442	green cha	3764	13.00159	95.22848	140.5903	115.3202	blue chan	-	-	-	-	-	colocalize	948	95.4346	86.02426	135.4436	104.7654
14	20231207_GB2_CON	V2 Vglut1	red chann	2171	26.4795	86.83141	113.7029	96.91419	green cha	3771	13.16256	94.28666	140.6675	114.4954	blue chan	-	-	-	-	-	colocalize	877	67.48005	89.97834	140.8974	109.6381
15	20231207_GB2_CON	V2 Vglut1	red chann	2815	31.02877	83.76092	111.8409	94.10443	green cha	3470	12.73343	97.47435	144.5323	118.2855	blue chan	-	-	-	-	-	colocalize	1078	102.6818	90.2449	142.6262	110.3149
16	20231207_GB2_CON	V2 Vglut1	red chann	2289	24.58628	96.94495	125.7348	107.9361	green cha	3323	13.41168	104.8089	157.7707	129.4125	blue chan	-	-	-	-	-	colocalize	850	59.15588	100.3076	154.1859	122.8207
17	20231207_GB2_CON	V2 Vglut1	red chann	2490	29.79277	81.76667	110.5133	92.4083	green cha	4571	14.37694	91.76504	140.4402	113.3175	blue chan	-	-	-	-	-	colocalize	1238	77.03675	86.23829	136.5202	106.1522
18	20231207_GB2_CON	V2 Vglut1	red chann	2022	21.89862	95.0178	123.5208	106.1856	green cha	4885	14.70235	82.31013	132.7415	103.2103	blue chan	-	-	-	-	-	colocalize	1021	40.55387	87.98286	137.9216	107.5806
19	20231207_GB2_CON	V2 Vglut1	red chann	2464	22.44602	95.01664	123.6644	106.2164	green cha	3348	11.74044	95.94325	145.3011	117.5251	blue chan	-	-	-	-	-	colocalize	863	43.00521	94.88644	144.2677	114.9889
20	20231207_GB2_CON	V2 Vglut1	red chann	2419	21.97106	101.0682	131.394	112.9485	green cha	3691	12.53427	90.7158	140.5733	112.0373	blue chan	-	-	-	-	-	colocalize	940	32.77872	95.32606	145.6319	115.4419
21	20231207_GB2_CON	V2 Vglut1	red chann	2581	22.7408	104.9826	136.2115	117.3121	green cha	3984	14.02209	90.52385	144.5	113.5588	blue chan	-	-	-	-	-	colocalize	1096	35.53102	97.13458	150.5671	118.671
22	20231207_GB2_CON	V2 Vglut1	red chann	2767	25.56198	87.9082	119.5703	100.0213	green cha	3821	12.7375	99.02277	147.9275	121.2907	blue chan	-	-	-	-	-	colocalize	1089	40.95133	92.82782	143.8586	113.8806
23	20231207_GB2_CON	V2 Vglut1	red chann	2308	21.01776	96.12565	127.4211	108.5589	green cha	3629	12.07936	96.05704	146.3582	118.4292	blue chan	-	-	-	-	-	colocalize	858	31.50583	95.54371	146.398	116.6679
24	20231207_GB2_CON	V2 Vglut1	red chann	3090	24.75372	91.9356	123.8683	104.2437	green cha	3901	12.64958	91.79544	142.3238	113.3998	blue chan	-	-	-	-	-	colocalize	1244	39.18931	91.33802	143.1969	111.7976
25	20231207_GB2_CON	V2 Vglut1	red chann	2657	23.90064	101.1118	132.9116	113.7301	green cha	4000	13.256	88.5155	138.8013	109.8164	blue chan	-	-	-	-	-	colocalize	1179	40.57379	94.19678	145.8346	114.7743
26	20231207_GB2_CON	V2 Vglut1	red chann	2324	20.58692	111.2169	141.7336	123.9173	green cha	3366	13.01456	94.53892	145.8099	116.757	blue chan	-	-	-	-	-	colocalize	775	35.45226	102.1935	153.4639	123.8633
27	20231207_GB2_CON	V2 Vglut1	red chann	2576	26.76009	90.97943	122.9984	103.2508	green cha	4526	12.50663	101.3467	147.6684	123.0297	blue chan	-	-	-	-	-	colocalize	1361	70.96657	95.44967	144.8894	115.9664
28	20231207_GB2_CON	V2 Vglut1	red chann	2706	25.30155	91.9963	123.912	104.2655	green cha	3097	10.31805	105.8389	152.7627	127.5166	blue chan	-	-	-	-	-	colocalize	934	43.92024	98.11777	148.8978	118.9338
29	20231207_GB2_CON	V2 Vglut1	red chann	2082	19.19308	107.1724	138.3381	119.9697	green cha	2533	9.556652	107.7967	154.3439	129.3113	blue chan	-	-	-	-	-	colocalize	646	31.13003	106.8127	154.3011	127.2173
30	20231207_GB2_CON	V2 Vglut1	red chann	2635	22.97875	104.1309	136.8095	117.3413	green cha	3579	10.8779	93.86588	141.6384	114.2227	blue chan	-	-	-	-	-	colocalize	1040	39.75144	98.31154	149.2913	118.7153
31	20231207_GB2_CON	V2 Vglut1	red chann	2253	19.21172	119.4021	150.9783	133.1638	green cha	3655	11.79781	93.68974	145.4055	115.737	blue chan	-	-	-	-	-	colocalize	1010	28.42772	105.8906	155.5787	126.8967
32	20231207_GB2_CON	V2 Vglut1	red chann	2342	24.13194	77.66311	100.6763	86.1595	green cha	3375	11.2797	75.08207	119.2797	93.11575	blue chan	-	-	-	-	-	colocalize	812	45.09052	75.79249	120.2414	91.84466
33	20231207_GB2_CON	V2 Vglut1	red chann	2650	23.56302	81.71509	106.0721	90.8311	green cha	3602	11.64881	67.55552	108.3917	83.81298	blue chan	-	-	-	-	-	colocalize	952	42.22742	74.19538	118.8451	90.00967
34	20231207_GB2_CON	V2 Vglut1	red chann	2112	18.991	94.92945	120.1965	104.6958	green cha	3118	12.52213	69.49455	112.533	86.68235	blue chan	-	-	-	-	-	colocalize	686	32.54519	81.80466	127.777	98.58117
35	20231207_GB2_CON	V2 Vglut1	red chann	2383	19.86068	94.94461	121.2707	105.0746	green cha	5081	16.08857	68.18303	138.3399	97.18363	blue chan	-	-	-	-	-	colocalize	1215	33.65885	80.90617	142.4868	104.6571
36	20231207_GB2_CON	V2 Vglut1	red chann	2015	21.0536	70.78462	93.45806	79.32324	green cha	3592	13.11442	86.58018	172.936	126.4422	blue chan	-	-	-	-	-	colocalize	938	30.72388	77.97548	141.662	106.0384
37	20231207_GB2_CON	V2 Vglut1	red chann	2336	28.7256	62.53082	82.53896	69.7377	green cha	3606	11.67194	81.16084	125.8741	99.63486	blue chan	-	-	-	-	-	colocalize	807	90.57373	71.42379	116.5694	87.36234

Loading Tabular Data

~~~
```{r load data}

data_cleaned_1 <- read_csv("Vamp2 Vglut1 and Psd95.csv")
data_cleaned_2 <- read_csv("Vamp2-Vglut1 Psd95-2.csv")
data_cleaned_3 <- read_csv("Vamp2-glut1-psd95-3.csv")
~~~

Tidy Data



> "TIDY DATA is a standard way of mapping the meaning of a dataset to its structure."
>
> —HADLEY WICKHAM

In tidy data:

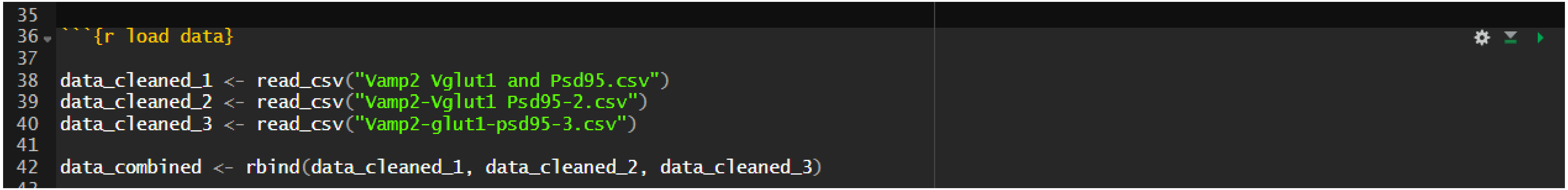
- each variable forms a column
- each observation forms a row
- each cell is a single measurement

id	name	color
1	floof	gray
2	max	black
3	cat	orange
4	donut	gray
5	merlin	black
6	panda	calico

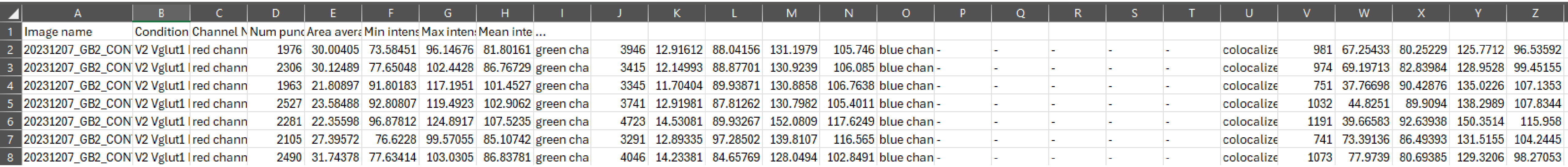
Wickham, H. (2014). Tidy Data. Journal of Statistical Software *59* (10). DOI: 10.18637/jss.v059.i10

Slide from https://earobinson95.github.io/stat331-calpoly-text/02-tidy-data-and-basics-of-graphics.html

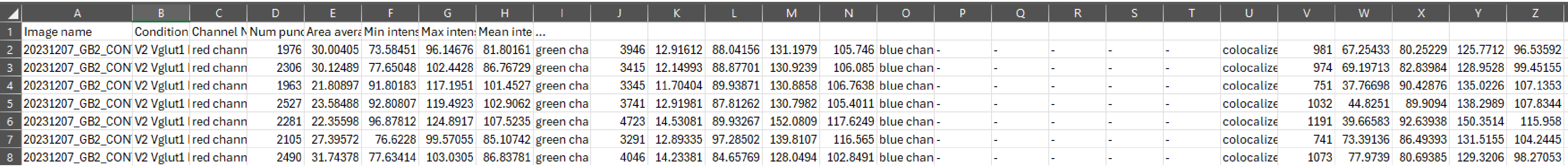
Combining Data with rbind



~~~

```{r load data}

data_cleaned_1 <- read_csv("Vamp2 Vglut1 and Psd95.csv")
data_cleaned_2 <- read_csv("Vamp2-Vglut1 Psd95-2.csv")
data_cleaned_3 <- read_csv("Vamp2-glut1-psd95-3.csv")

data_combined <- rbind(data_cleaned_1, data_cleaned_2, data_cleaned_3)
~~~

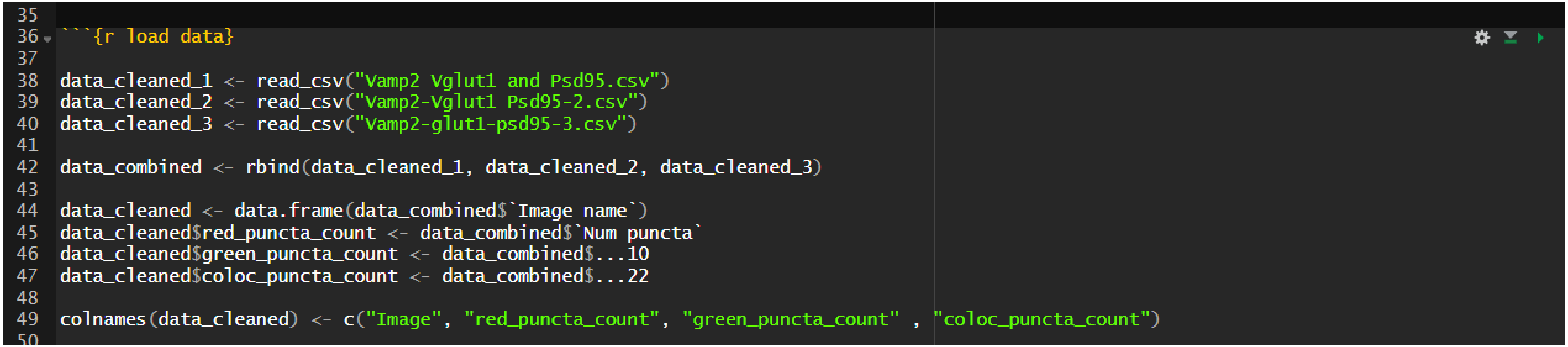


| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Image name | Condition | Channel N | Num punc | Area avera | Min intens | Max inten: | Mean inte | ... | | | | | | | | | | | | | | | | | |
| 2 | 20231207_GB2_CON | V2 Vglut1 I | red chann | 1976 | 30.00405 | 73.58451 | 96.14676 | 81.80161 | green cha | 3946 | 12.91612 | 88.04156 | 131.1979 | 105.746 | blue chan | - | - | - | - | - | colocalize | 981 | 67.25433 | 80.25229 | 125.7712 | 96.53592 |
| 3 | 20231207_GB2_CON | V2 Vglut1 I | red chann | 2306 | 30.12489 | 77.65048 | 102.4428 | 86.76729 | green cha | 3415 | 12.14993 | 88.87701 | 130.9239 | 106.085 | blue chan | - | - | - | - | - | colocalize | 974 | 69.19713 | 82.83984 | 128.9528 | 99.45155 |
| 4 | 20231207_GB2_CON | V2 Vglut1 I | red chann | 1963 | 21.80897 | 91.80183 | 117.1951 | 101.4527 | green cha | 3345 | 11.70404 | 89.93871 | 130.8858 | 106.7638 | blue chan | - | - | - | - | - | colocalize | 751 | 37.76698 | 90.42876 | 135.0226 | 107.1353 |
| 5 | 20231207_GB2_CON | V2 Vglut1 I | red chann | 2527 | 23.58488 | 92.80807 | 119.4923 | 102.9062 | green cha | 3741 | 12.91981 | 87.81262 | 130.7982 | 105.4011 | blue chan | - | - | - | - | - | colocalize | 1032 | 44.8251 | 89.9094 | 138.2989 | 107.8344 |
| 6 | 20231207_GB2_CON | V2 Vglut1 I | red chann | 2281 | 22.35598 | 96.87812 | 124.8917 | 107.5235 | green cha | 4723 | 14.53081 | 89.93267 | 152.0809 | 117.6249 | blue chan | - | - | - | - | - | colocalize | 1191 | 39.66583 | 92.63938 | 150.3514 | 115.958 |
| 7 | 20231207_GB2_CON | V2 Vglut1 I | red chann | 2105 | 27.39572 | 76.6228 | 99.57055 | 85.10742 | green cha | 3291 | 12.89335 | 97.28502 | 139.8107 | 116.565 | blue chan | - | - | - | - | - | colocalize | 741 | 73.39136 | 86.49393 | 131.5155 | 104.2445 |
| 8 | 20231207_GB2_CON | V2 Vglut1 I | red chann | 2490 | 31.74378 | 77.63414 | 103.0305 | 86.83781 | green cha | 4046 | 14.23381 | 84.65769 | 128.0494 | 102.8491 | blue chan | - | - | - | - | - | colocalize | 1073 | 77.9739 | 80.69385 | 129.3206 | 98.27053 |



| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Image name | Condition | Channel N | Num punc | Area avera | Min intens | Max inten: | Mean inte | ... | | | | | | | | | | | | | | | | | |
| 2 | 20231207_GB2_CON | V2 Vglut1 I | red chann | 1976 | 30.00405 | 73.58451 | 96.14676 | 81.80161 | green cha | 3946 | 12.91612 | 88.04156 | 131.1979 | 105.746 | blue chan | - | - | - | - | - | colocalize | 981 | 67.25433 | 80.25229 | 125.7712 | 96.53592 |
| 3 | 20231207_GB2_CON | V2 Vglut1 I | red chann | 2306 | 30.12489 | 77.65048 | 102.4428 | 86.76729 | green cha | 3415 | 12.14993 | 88.87701 | 130.9239 | 106.085 | blue chan | - | - | - | - | - | colocalize | 974 | 69.19713 | 82.83984 | 128.9528 | 99.45155 |
| 4 | 20231207_GB2_CON | V2 Vglut1 I | red chann | 1963 | 21.80897 | 91.80183 | 117.1951 | 101.4527 | green cha | 3345 | 11.70404 | 89.93871 | 130.8858 | 106.7638 | blue chan | - | - | - | - | - | colocalize | 751 | 37.76698 | 90.42876 | 135.0226 | 107.1353 |
| 5 | 20231207_GB2_CON | V2 Vglut1 I | red chann | 2527 | 23.58488 | 92.80807 | 119.4923 | 102.9062 | green cha | 3741 | 12.91981 | 87.81262 | 130.7982 | 105.4011 | blue chan | - | - | - | - | - | colocalize | 1032 | 44.8251 | 89.9094 | 138.2989 | 107.8344 |
| 6 | 20231207_GB2_CON | V2 Vglut1 I | red chann | 2281 | 22.35598 | 96.87812 | 124.8917 | 107.5235 | green cha | 4723 | 14.53081 | 89.93267 | 152.0809 | 117.6249 | blue chan | - | - | - | - | - | colocalize | 1191 | 39.66583 | 92.63938 | 150.3514 | 115.958 |
| 7 | 20231207_GB2_CON | V2 Vglut1 I | red chann | 2105 | 27.39572 | 76.6228 | 99.57055 | 85.10742 | green cha | 3291 | 12.89335 | 97.28502 | 139.8107 | 116.565 | blue chan | - | - | - | - | - | colocalize | 741 | 73.39136 | 86.49393 | 131.5155 | 104.2445 |
| 8 | 20231207_GB2_CON | V2 Vglut1 I | red chann | 2490 | 31.74378 | 77.63414 | 103.0305 | 86.83781 | green cha | 4046 | 14.23381 | 84.65769 | 128.0494 | 102.8491 | blue chan | - | - | - | - | - | colocalize | 1073 | 77.9739 | 80.69385 | 129.3206 | 98.27053 |

# Extract Columns of Interest



~~~
```{r load data}

data_cleaned_1 <- read_csv("Vamp2 Vglut1 and Psd95.csv")
data_cleaned_2 <- read_csv("Vamp2-Vglut1 Psd95-2.csv")
data_cleaned_3 <- read_csv("Vamp2-glut1-psd95-3.csv")

data_combined <- rbind(data_cleaned_1, data_cleaned_2, data_cleaned_3)

data_cleaned <- data.frame(data_combined$`Image name`)
data_cleaned$red_puncta_count <- data_combined$`Num puncta`
data_cleaned$green_puncta_count <- data_combined$...10
data_cleaned$coloc_puncta_count <- data_combined$...22

colnames(data_cleaned) <- c("Image", "red_puncta_count", "green_puncta_count" , "coloc_puncta_count")
~~~

# Get condition from image name

~~~
```{r load data}

data_cleaned_1 <- read_csv("Vamp2 Vglut1 and Psd95.csv")
data_cleaned_2 <- read_csv("Vamp2-Vglut1 Psd95-2.csv")
data_cleaned_3 <- read_csv("Vamp2-glut1-psd95-3.csv")

data_combined <- rbind(data_cleaned_1, data_cleaned_2, data_cleaned_3)

data_cleaned <- data.frame(data_combined$`Image name`)
data_cleaned$red_puncta_count <- data_combined$`Num puncta`
data_cleaned$green_puncta_count <- data_combined$...10
data_cleaned$coloc_puncta_count <- data_combined$...22

colnames(data_cleaned) <- c("Image", "red_puncta_count", "green_puncta_count" , "coloc_puncta_count")

#initialize condition
data_cleaned$condition <- "control"

#label data with experimental condition
data_cleaned[which(grepl("*CONTROL*", data_cleaned$Image)),]$condition <- "control"
data_cleaned[which(grepl("*VAMP2 TRAP*", data_cleaned$Image)),]$condition <- "Vamp2 Trap"

#make condition a factor so that it always plots in control then VAMP2 TRAP order
data_cleaned$condition <- factor(data_cleaned$condition, levels = c("control", "Vamp2 Trap"))
```
~~~

Get condition from image name

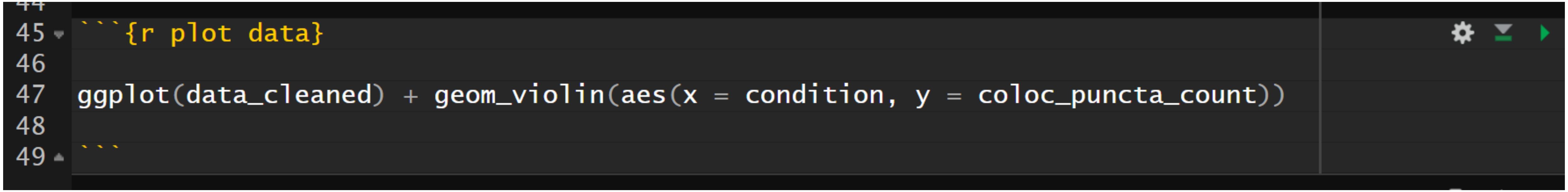
~~~
```{r load data}

data_cleaned_1 <- read_csv("Vamp2 Vglut1 and Psd95.csv")
data_cleaned_2 <- read_csv("Vamp2-Vglut1 Psd95-2.csv")
data_cleaned_3 <- read_csv("Vamp2-glut1-psd95-3.csv")

data_combined <- rbind(data_cleaned_1, data_cleaned_2, data_cleaned_3)

data_cleaned <- data.frame(data_combined$`Image name`)
data_cleaned$red_puncta_count <- data_combined$`Num puncta`
data_cleaned$green_puncta_count <- data_combined$...10
data_cleaned$coloc_puncta_count <- data_combined$...22

colnames(data_cleaned) <- c("Image", "red_puncta_count", "green_puncta_count" , "coloc_punct

#initialize condition
data_cleaned$condition <- "control"

#label data with experimental condition
data_cleaned[which(grepl("*CONTROL*", data_cleaned$Image)),]$condition <- "control"
data_cleaned[which(grepl("*VAMP2 TRAP*", data_cleaned$Image)),]$condition <- "Vamp2 Trap"

#make condition a factor so that it always plots in control then VAMP2 TRAP order
data_cleaned$condition <- factor(data_cleaned$condition, levels = c("control", "Vamp2 Trap"))
```
~~~

each column a variable

id	name	color
1	floof	gray
2	max	black
3	cat	orange
4	donut	gray
5	merlin	black
6	panda	calico

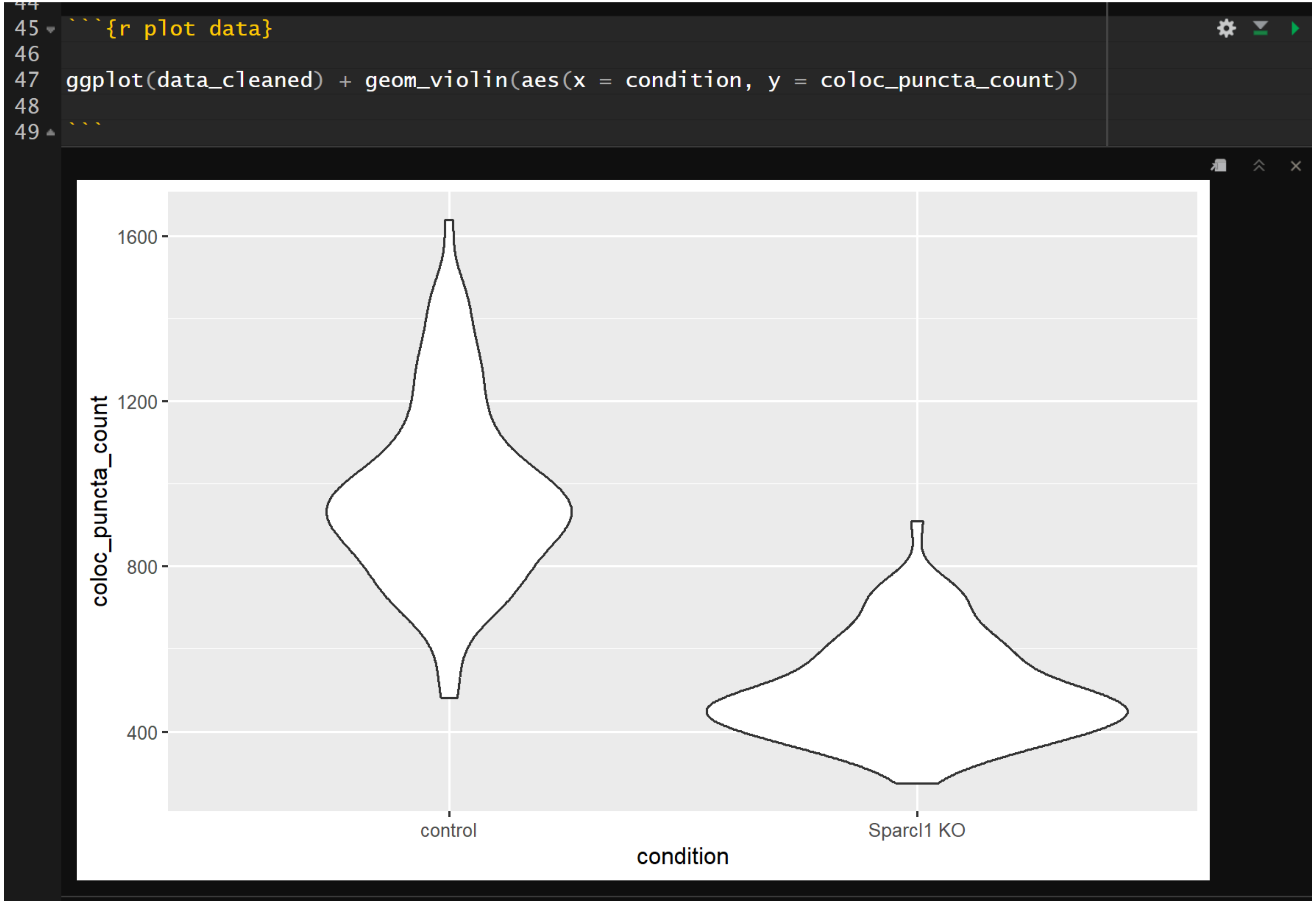
each row an observation

Simple plots with ggplot



~~~
```{r plot data}

ggplot(data_cleaned) + geom_violin(aes(x = condition, y = coloc_puncta_count))

~~~
```
```

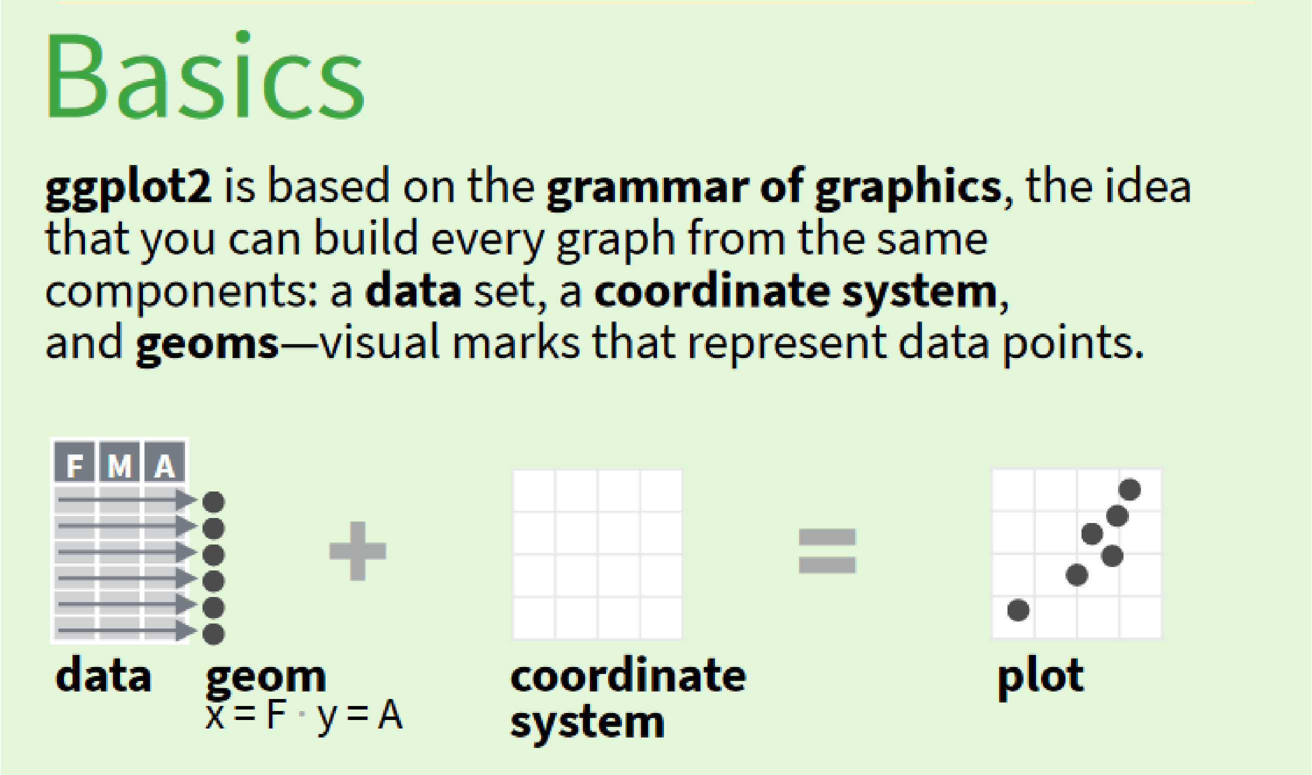
Simple plots with ggplot



~~~
```{r plot data}

ggplot(data_cleaned) + geom_violin(aes(x = condition, y = coloc_puncta_count))

~~~
```
```

Ggplot cheatsheet

- https://rstudio.github.io/cheatsheets/html/data-visualization.html



Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and **geoms**—visual marks that represent data points.

Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

GRAPHICAL PRIMITIVES

a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))

a + geom_blank() and **a + expand_limits()**
Ensure limits include values across all plots.

b + geom_curve(aes(yend = lat + 1, xend = long + 1), curvature = 1**)** - x, xend, y, yend, alpha, angle, color, curvature, linetype, size

a + geom_path(lineend = "butt", linejoin = "round", linemitre = 1**)**
x, y, alpha, color, group, linetype, size

a + geom_polygon(aes(alpha = 50)**)** - x, y, alpha, color, fill, group, subgroup, linetype, size

b + geom_rect(aes(xmin = long, ymin = lat, xmax = long + 1, ymax = lat + 1)**)** - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

a + geom_ribbon(aes(ymin = unemploy - 900, ymax = unemploy + 900)**)** - x, ymax, ymin, alpha, color, fill, group, linetype, size

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

b + geom_abline(aes(intercept = 0, slope = 1)**)**
b + geom_hline(aes(yintercept = lat)**)**
b + geom_vline(aes(xintercept = long)**)**

b + geom_segment(aes(yend = lat + 1, xend = long + 1)**)**
b + geom_spoke(aes(angle = 1:1155, radius = 1)**)**

ONE VARIABLE continuous

c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)

c + geom_area(stat = "bin"**)**
x, y, alpha, color, fill, linetype, size

c + geom_density(kernel = "gaussian"**)**
x, y, alpha, color, fill, group, linetype, size, weight

c + geom_dotplot()
x, y, alpha, color, fill

c + geom_freqpoly()
x, y, alpha, color, group, linetype, size

c + geom_histogram(binwidth = 5**)**
x, y, alpha, color, fill, linetype, size, weight

c2 + geom_qq(aes(sample = hwy)**)**
x, y, alpha, color, fill, linetype, size, weight

discrete

d <- ggplot(mpg, aes(fl))

d + geom_bar()
x, alpha, color, fill, linetype, size, weight

TWO VARIABLES

both continuous

e <- ggplot(mpg, aes(cty, hwy))

e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1**)** - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

e + geom_point()
x, y, alpha, color, fill, shape, size, stroke

e + geom_quantile()
x, y, alpha, color, group, linetype, size, weight

e + geom_rug(sides = “bl”**)**
x, y, alpha, color, linetype, size

e + geom_smooth(method = lm**)**
x, y, alpha, color, fill, group, linetype, size, weight

e + geom_text(aes(label = cty), nudge_x = 1, nudge_y = 1**)** - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

one discrete, one continuous

f <- ggplot(mpg, aes(class, hwy))

f + geom_col()
x, y, alpha, color, fill, group, linetype, size

f + geom_boxplot()
x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight

f + geom_dotplot(binaxis = "y", stackdir = “center”**)**
x, y, alpha, color, fill, group

f + geom_violin(scale = “area”**)**
x, y, alpha, color, fill, group, linetype, size, weight

both discrete

g <- ggplot(diamonds, aes(cut, color))

g + geom_count()
x, y, alpha, color, fill, shape, size, stroke

e + geom_jitter(height = 2, width = 2**)**
x, y, alpha, color, fill, shape, size

continuous bivariate distribution

h <- ggplot(diamonds, aes(carat, price))

h + geom_bin2d(binwidth = c(0.25, 500)**)**
x, y, alpha, color, fill, linetype, size, weight

h + geom_density_2d()
x, y, alpha, color, group, linetype, size

h + geom_hex()
x, y, alpha, color, fill, size

continuous function

i <- ggplot(economics, aes(date, unemploy))

i + geom_area()
x, y, alpha, color, fill, linetype, size

i + geom_line()
x, y, alpha, color, group, linetype, size

i + geom_step(direction = "hv"**)**
x, y, alpha, color, group, linetype, size

visualizing error

df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit - se, ymax = fit + se))

j + geom_crossbar(fatten = 2**)** - x, y, ymax, ymin, alpha, color, fill, group, linetype, size

j + geom_errorbar() - x, ymax, ymin, alpha, color, group, linetype, size, width
Also **geom_errorbarh()**.

j + geom_linerange()
x, ymin, ymax, alpha, color, group, linetype, size

j + geom_pointrange() - x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

maps

Draw the appropriate geometric object depending on the simple features present in the data. aes() arguments: map_id, alpha, color, fill, linetype, linewidth.

nc <- **sf::st_read(**system.file("shape/nc.shp", package = "sf")**)**

ggplot(nc) +
geom_sf(aes(fill = AREA)**)**

THREE VARIABLES

seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2)); l <- ggplot(seals, aes(long, lat))

l + geom_contour(aes(z = z)**)**
x, y, z, alpha, color, group, linetype, size, weight

l + geom_contour_filled(aes(fill = z)**)**
x, y, alpha, color, fill, group, linetype, size, subgroup

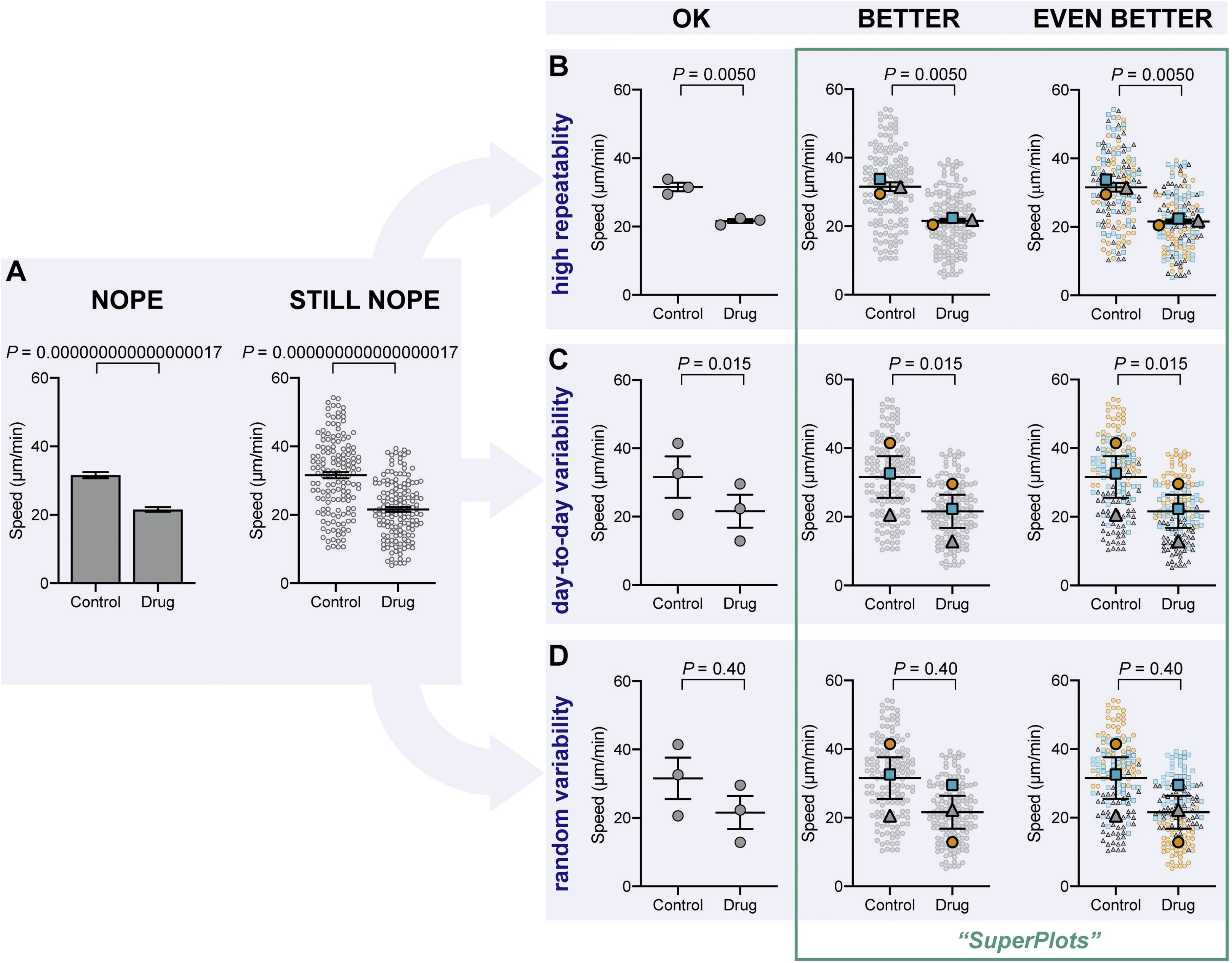
l + geom_raster(aes(fill = z), hjust = 0.5, vjust = 0.5, interpolate = FALSE**)**
x, y, alpha, fill

l + geom_tile(aes(fill = z)**)**
x, y, alpha, color, fill, linetype, size, width

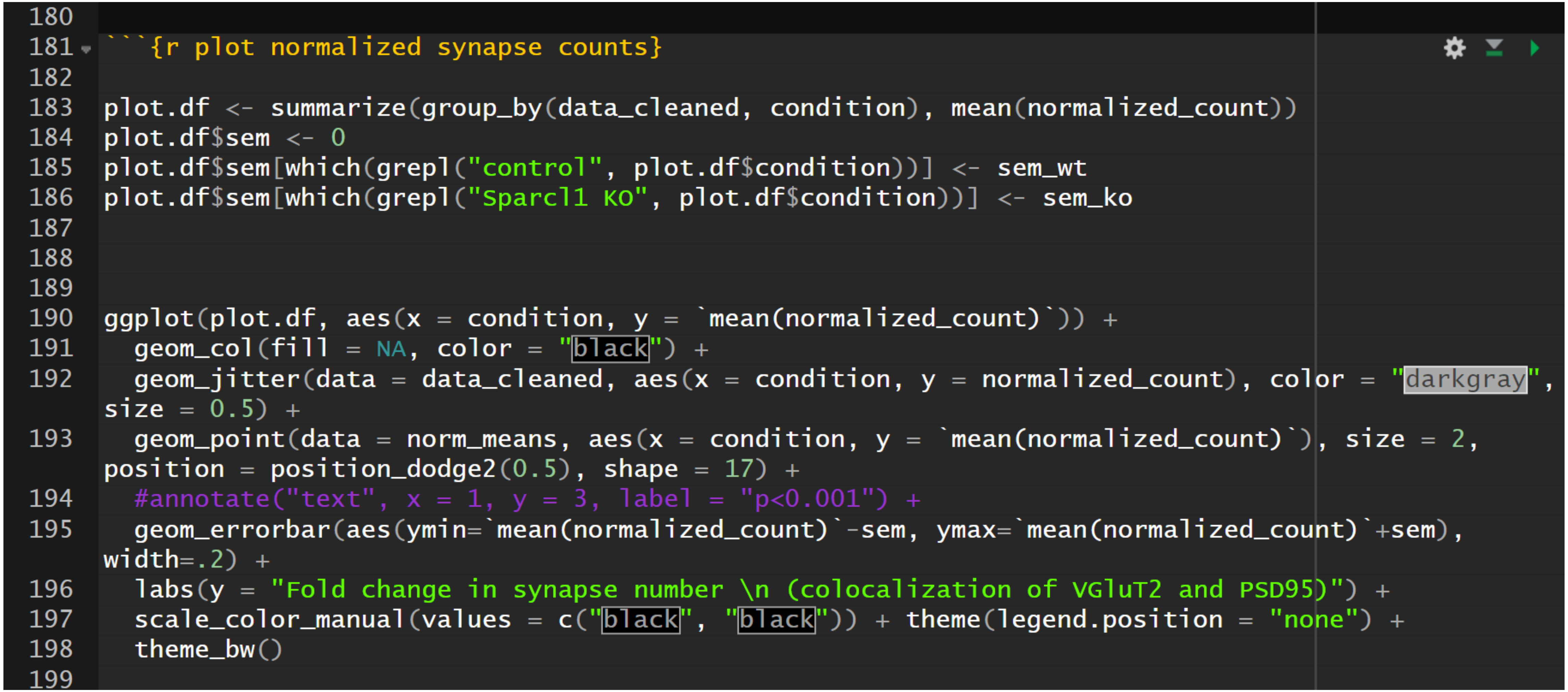
SuperPlots

- “We have written this tutorial to help cell biologists plot data in a way that highlights both experimental robustness and cell-to-cell variability. Specifically, we propose the use of distribution–reproducibility **“SuperPlots”** that display the distribution of the entire dataset, and report statistics (such as means, error bars, and P values) that address the reproducibility of the findings.”

https://doi.org/10.1083/jcb.202001064

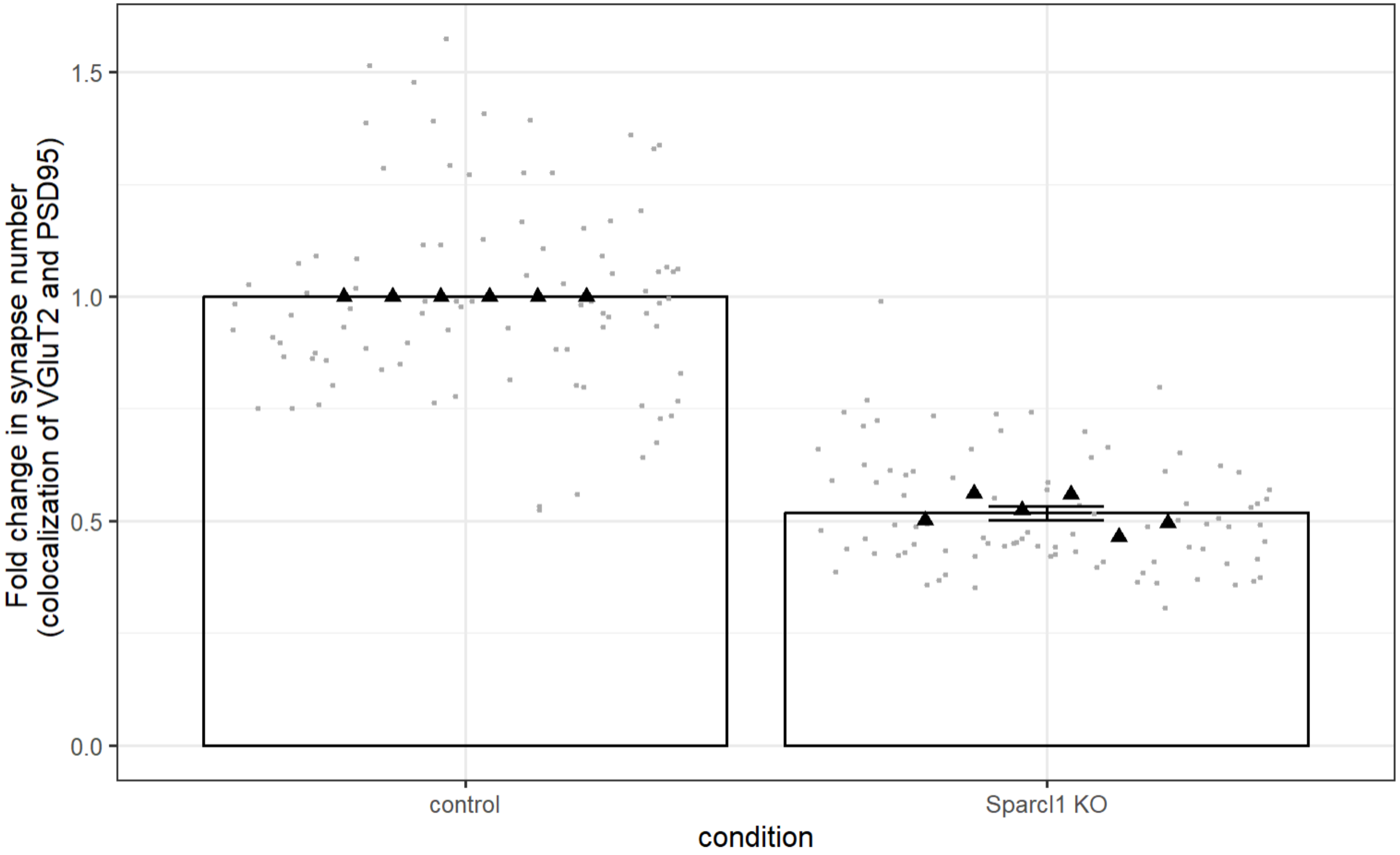

A
NOPE
STILL NOPE
P = 0.000000000000000017
P = 0.000000000000000017
Speed (µm/min)
0
20
40
60
Control
Drug
OK
BETTER
EVEN BETTER
B
high repeatability
P = 0.0050
C
day-to-day variability
P = 0.015
D
random variability
P = 0.40
"SuperPlots"

Multicomponent plot

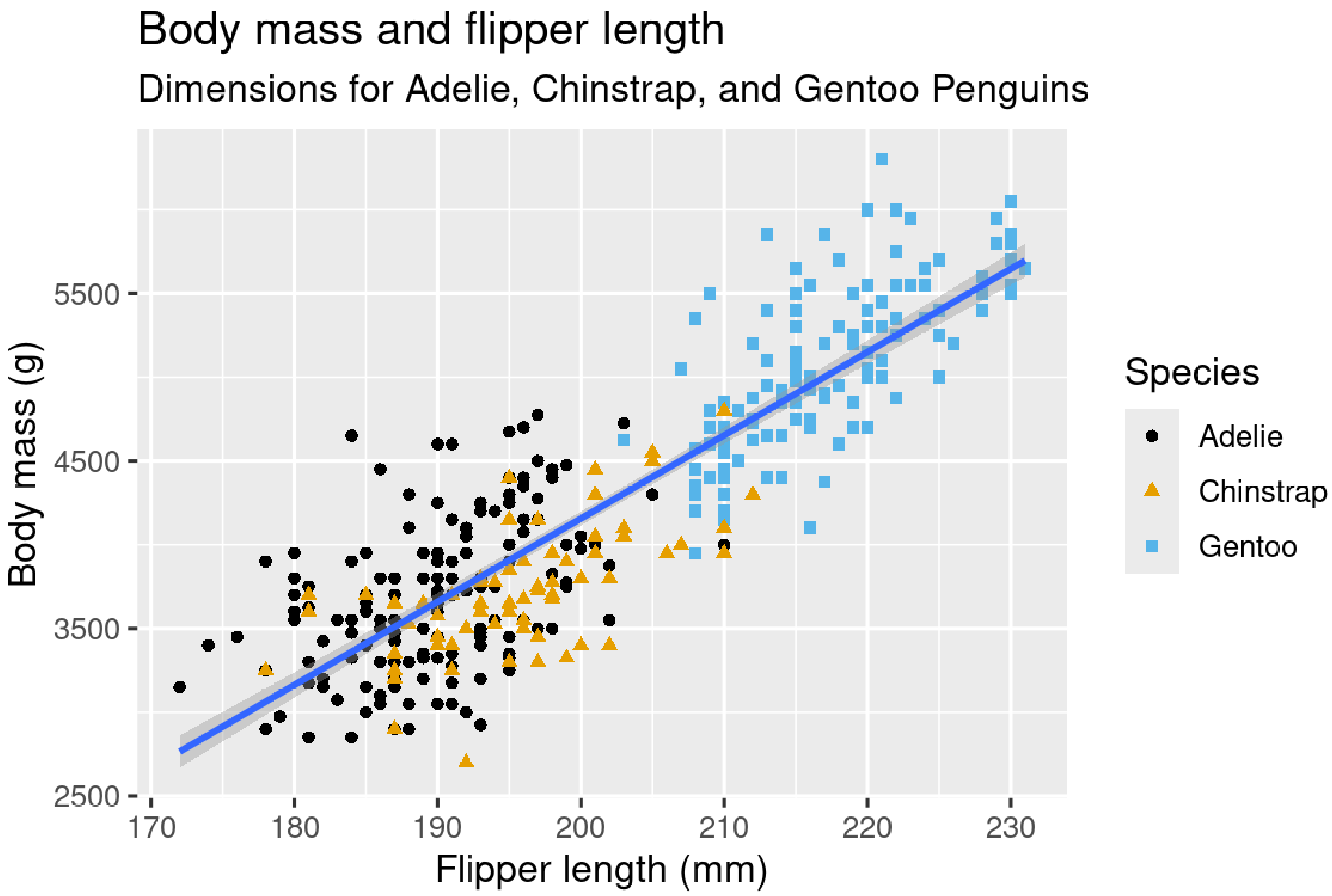


~~~

```{r plot normalized synapse counts}

plot.df <- summarize(group_by(data_cleaned, condition), mean(normalized_count))
plot.df$sem <- 0
plot.df$sem[which(grepl("control", plot.df$condition))] <- sem_wt
plot.df$sem[which(grepl("Sparcl1 KO", plot.df$condition))] <- sem_ko



ggplot(plot.df, aes(x = condition, y = `mean(normalized_count)`)) +
  geom_col(fill = NA, color = "black") +
  geom_jitter(data = data_cleaned, aes(x = condition, y = normalized_count), color = "darkgray", size = 0.5) +
  geom_point(data = norm_means, aes(x = condition, y = `mean(normalized_count)`), size = 2, position = position_dodge2(0.5), shape = 17) +
  #annotate("text", x = 1, y = 3, label = "p<0.001") +
  geom_errorbar(aes(ymin=`mean(normalized_count)`-sem, ymax=`mean(normalized_count)`+sem), width=.2) +
  labs(y = "Fold change in synapse number \n (colocalization of VGluT2 and PSD95)") +
  scale_color_manual(values = c("black", "black")) + theme(legend.position = "none") +
  theme_bw()

~~~

# Multicomponent plot

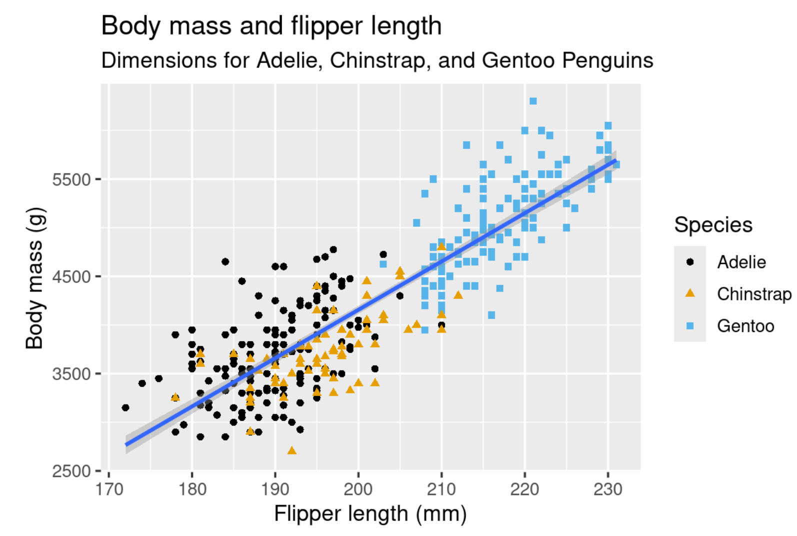


# Additional components



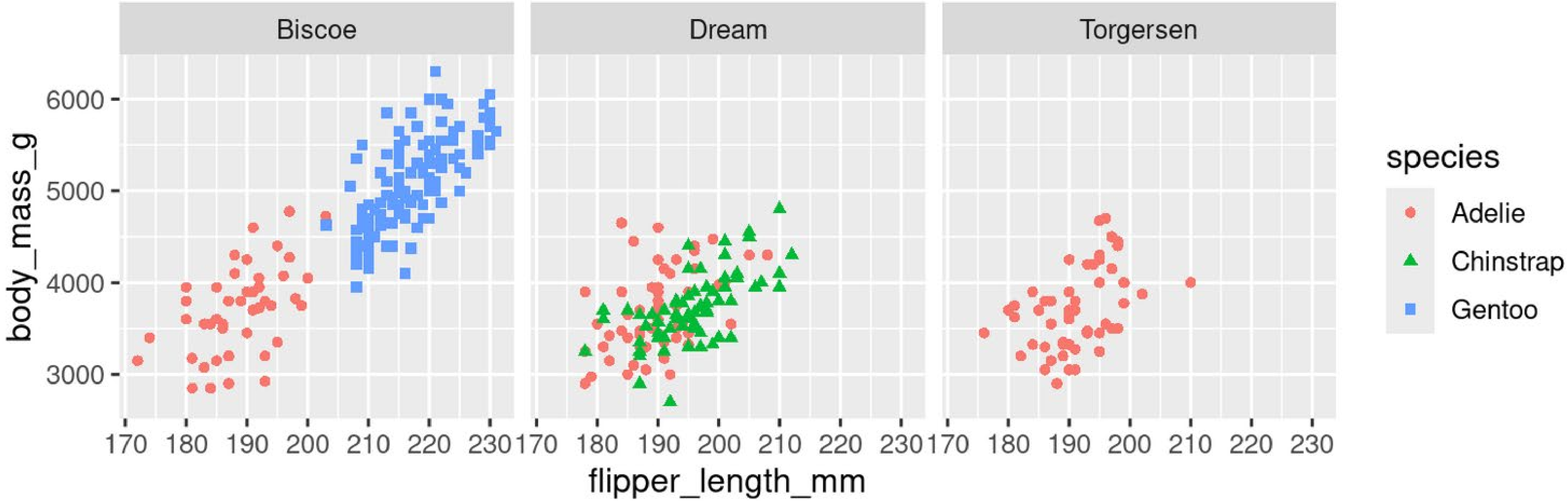
# Additional components

```
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm, y = body_mass_g)
) +
  geom_point(aes(color = species, shape = species)) +
  geom_smooth(method = "lm") +
  labs(
    title = "Body mass and flipper length",
    subtitle = "Dimensions for Adelie, Chinstrap, and Gentoo Penguins",
    x = "Flipper length (mm)", y = "Body mass (g)",
    color = "Species", shape = "Species"
) +
  scale_color_colorblind()
```



# Facet wrap

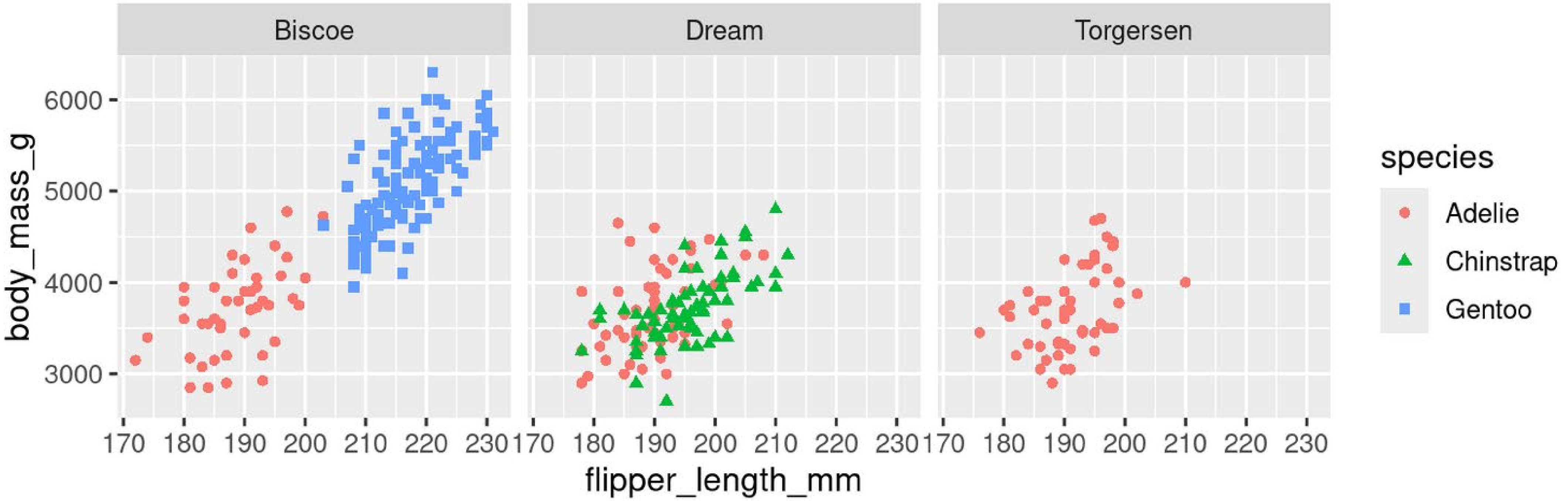
```r
ggplot(penguins, aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point(aes(color = species, shape = species)) +
  facet_wrap(~island)
```



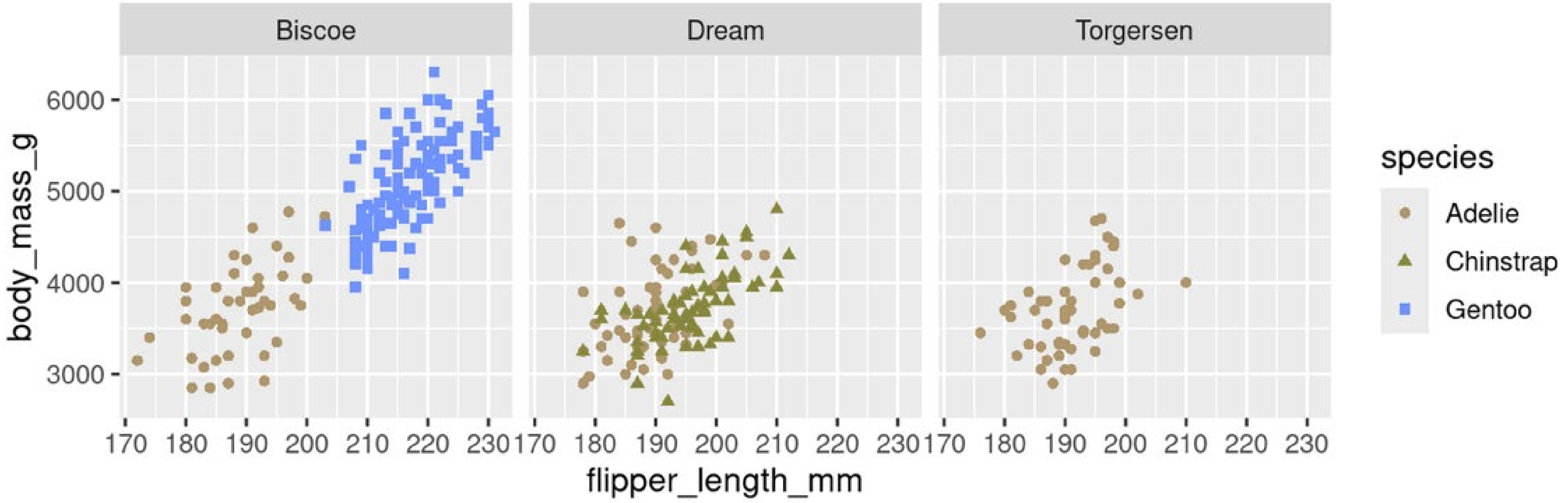
Accessible Color Palletes

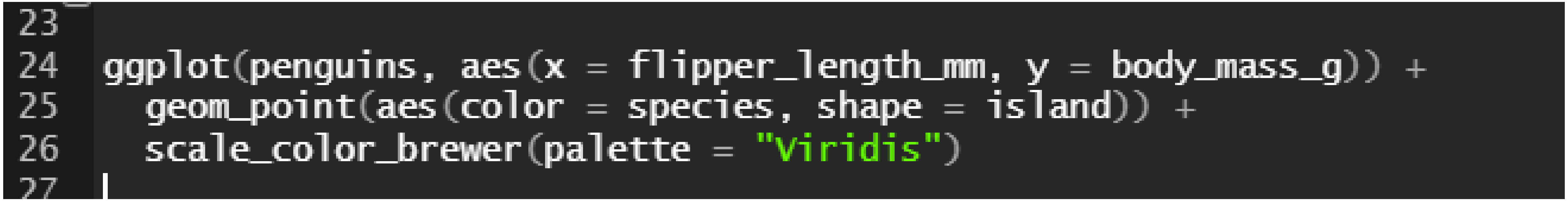
- Colorblindness affects ~4% of people mostly through red-green color blindness
- To allow these people to view our figures, red-green discriminations should be avoided
- Many packages are changing their default packages to be more color-blind accessible. Viridis and YlGnBu are good options.
- You can check how your image would appear to a person with color blindness using online apps like this one https://bioapps.byu.edu/colorblind_image_tester

Normal vision

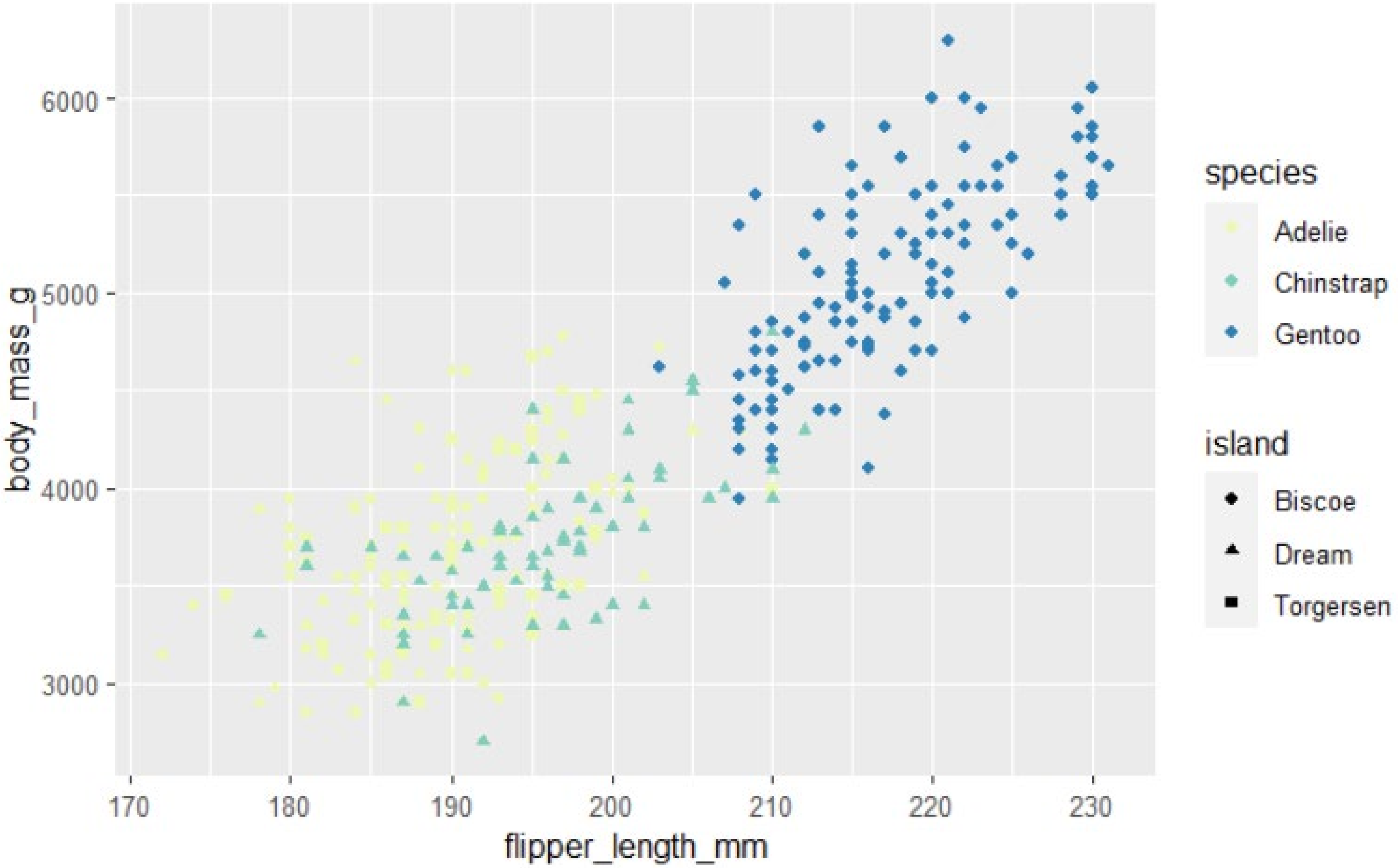


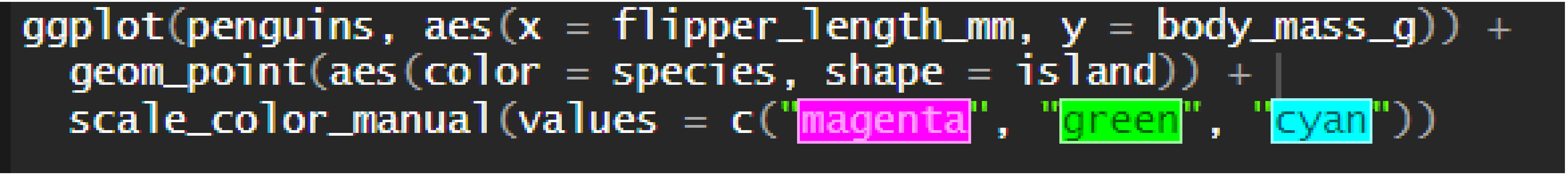
Color blind



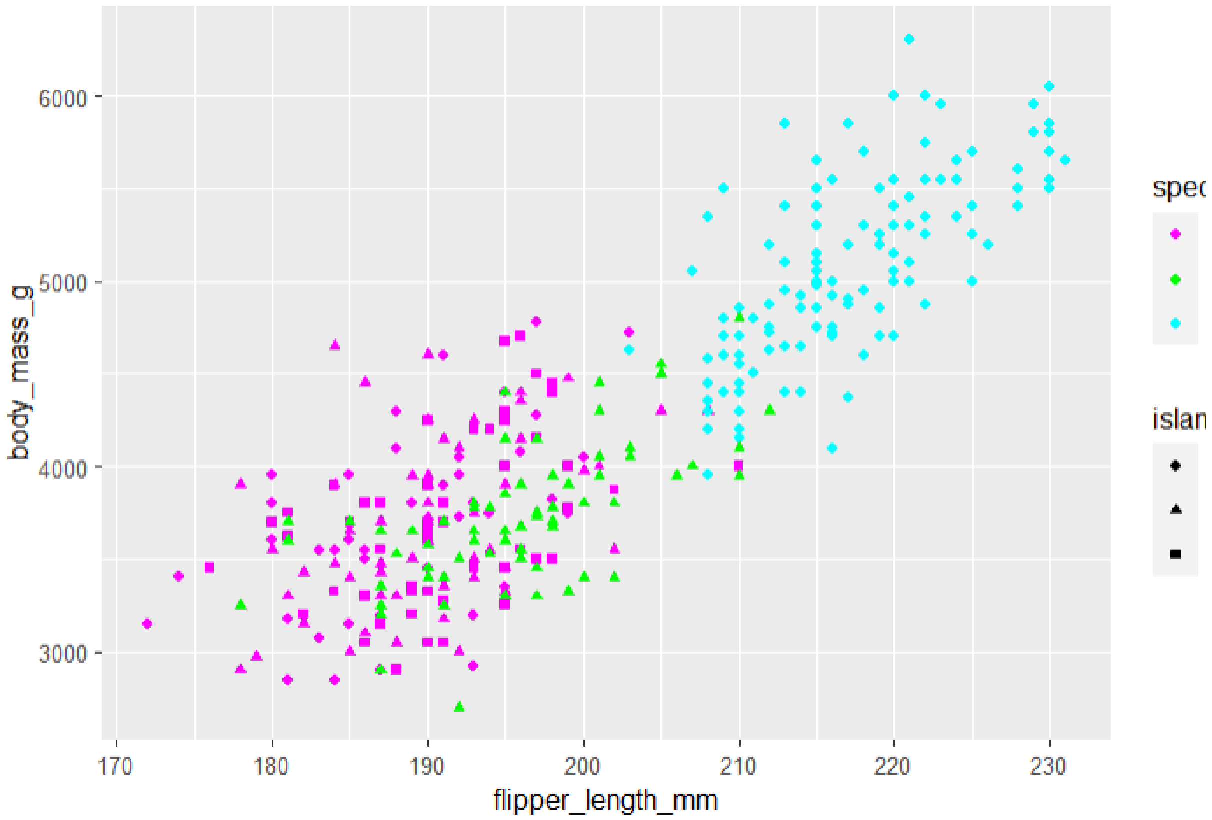


```r

ggplot(penguins, aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point(aes(color = species, shape = island)) +
  scale_color_brewer(palette = "Viridis")

```

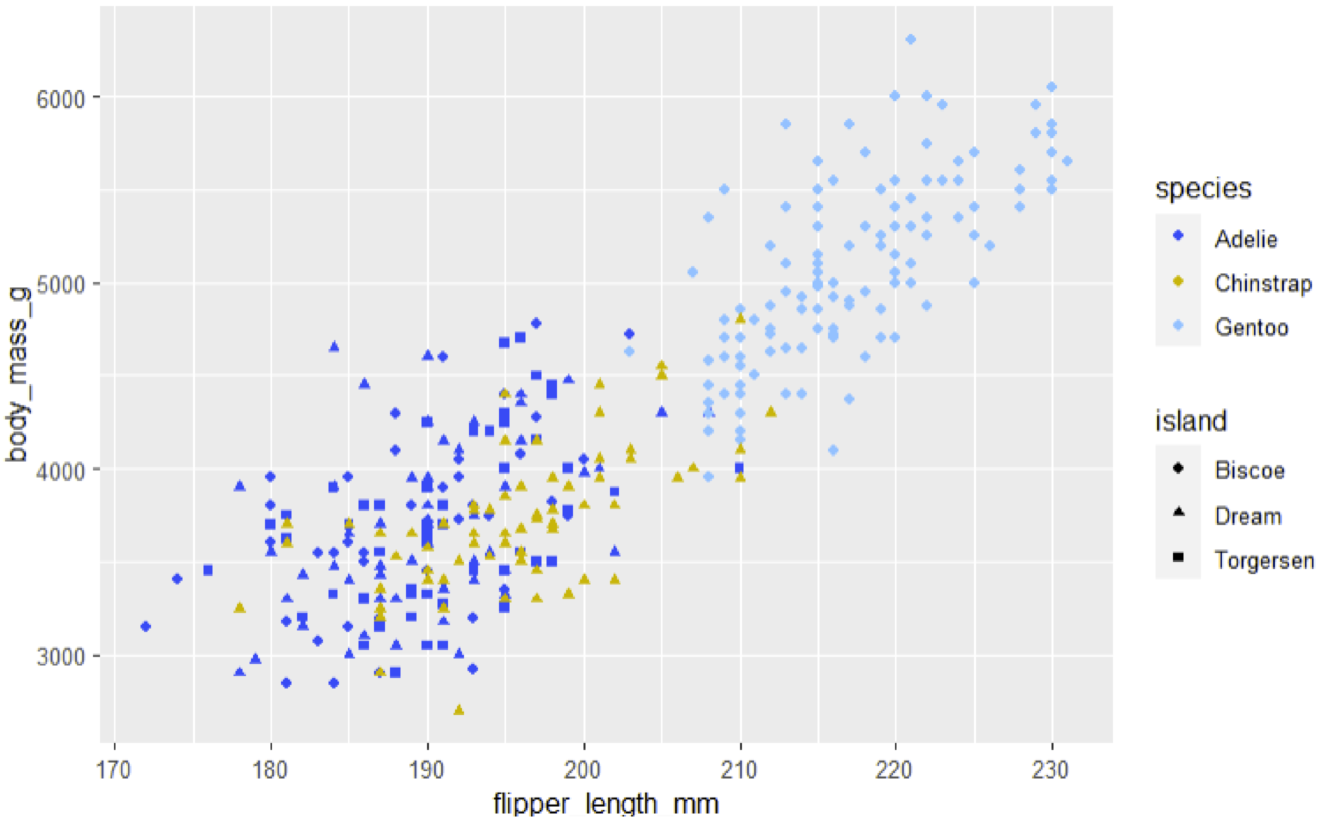




```
ggplot(penguins, aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point(aes(color = species, shape = island)) +
  scale_color_manual(values = c("magenta", "green", "cyan"))
```

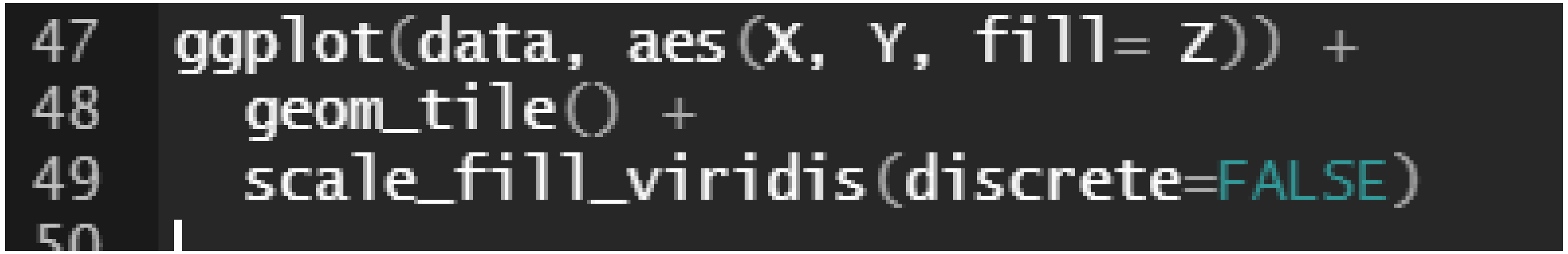


Normal vision

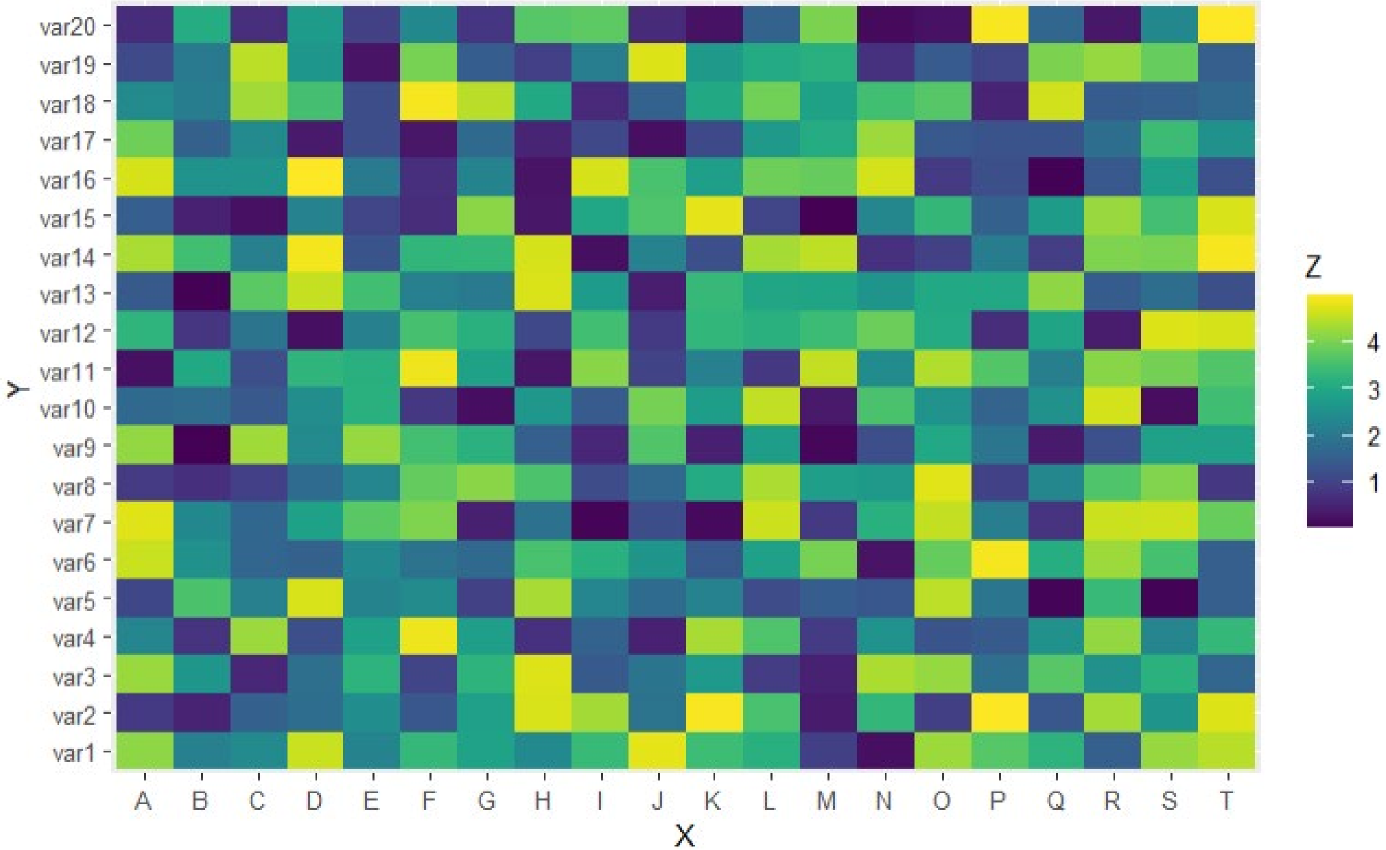


Color blind

Heatmaps

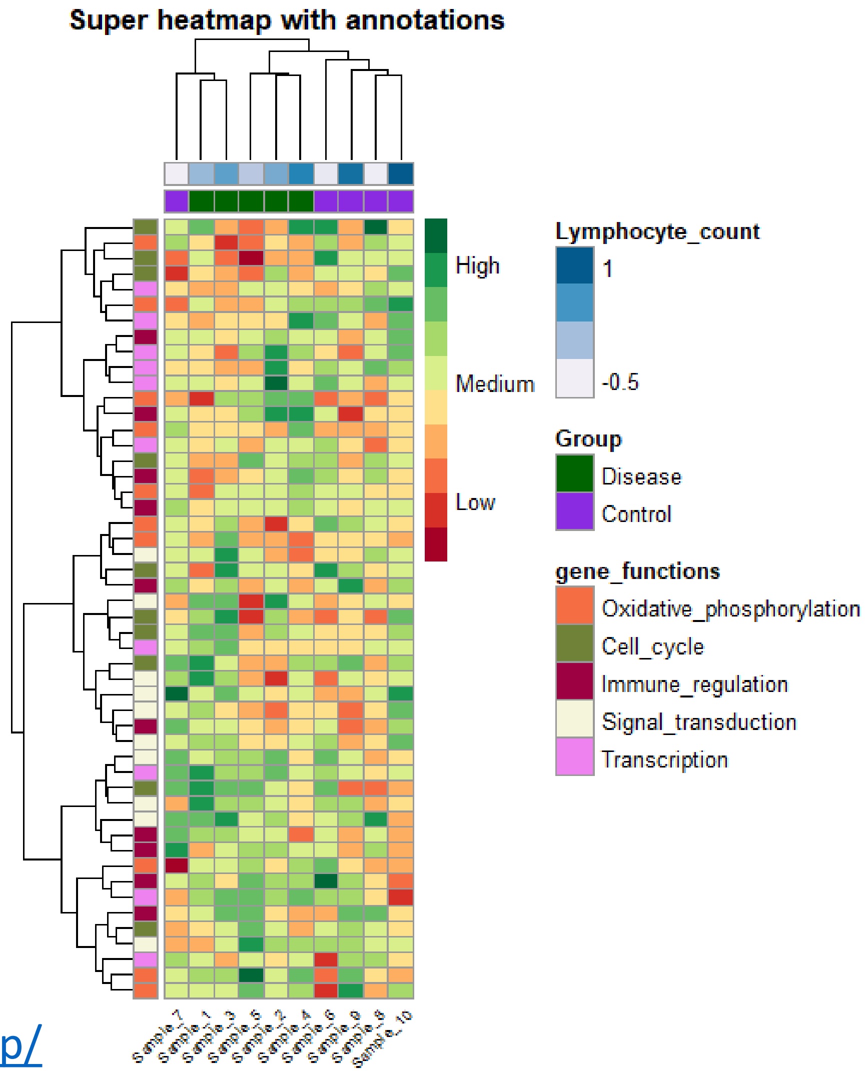


```
ggplot(data, aes(X, Y, fill= Z)) +
  geom_tile() +
  scale_fill_viridis(discrete=FALSE)
```



Heatmaps for complex data with pheatmap

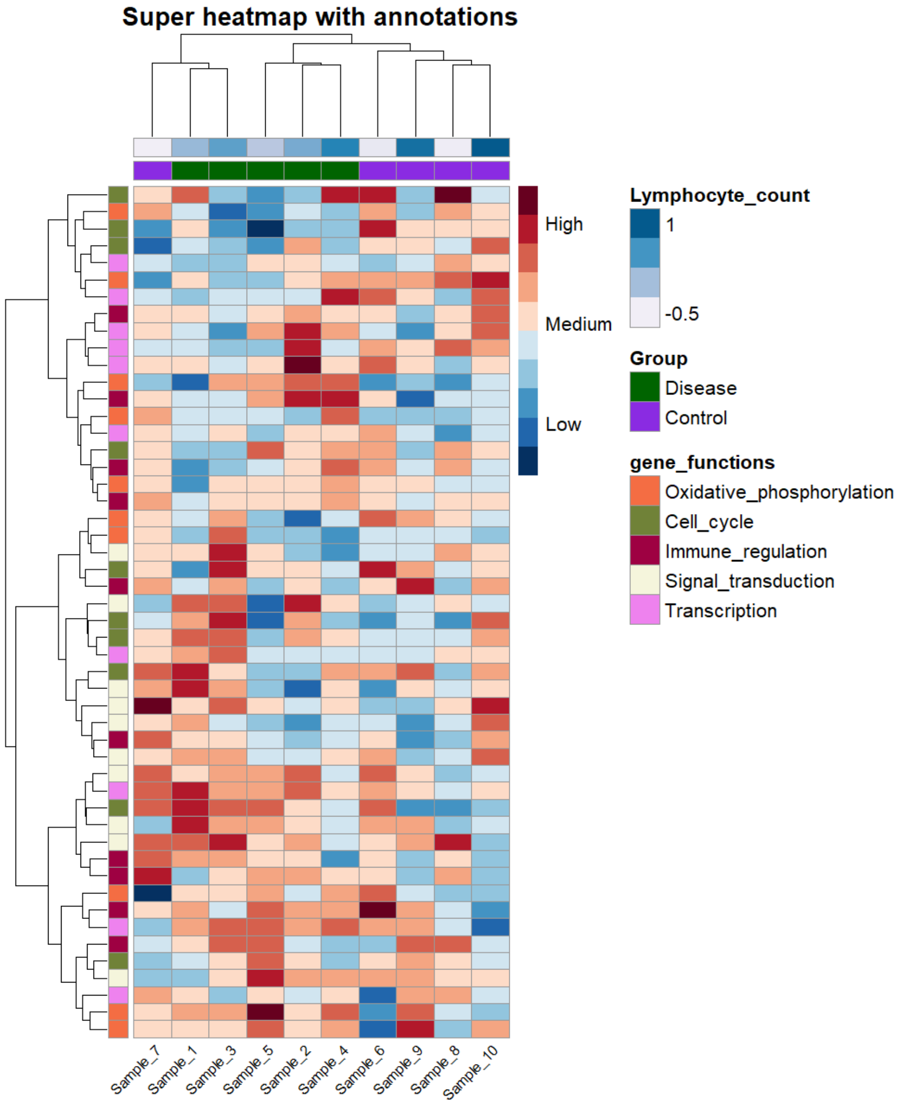
- Pheatmap can add clustering of your heatmap rows and columns
- Pheatmap makes additional annotations easier than ggplot



https://biostatsquid.com/step-by-step-heatmap-tutorial-with-pheatmap/

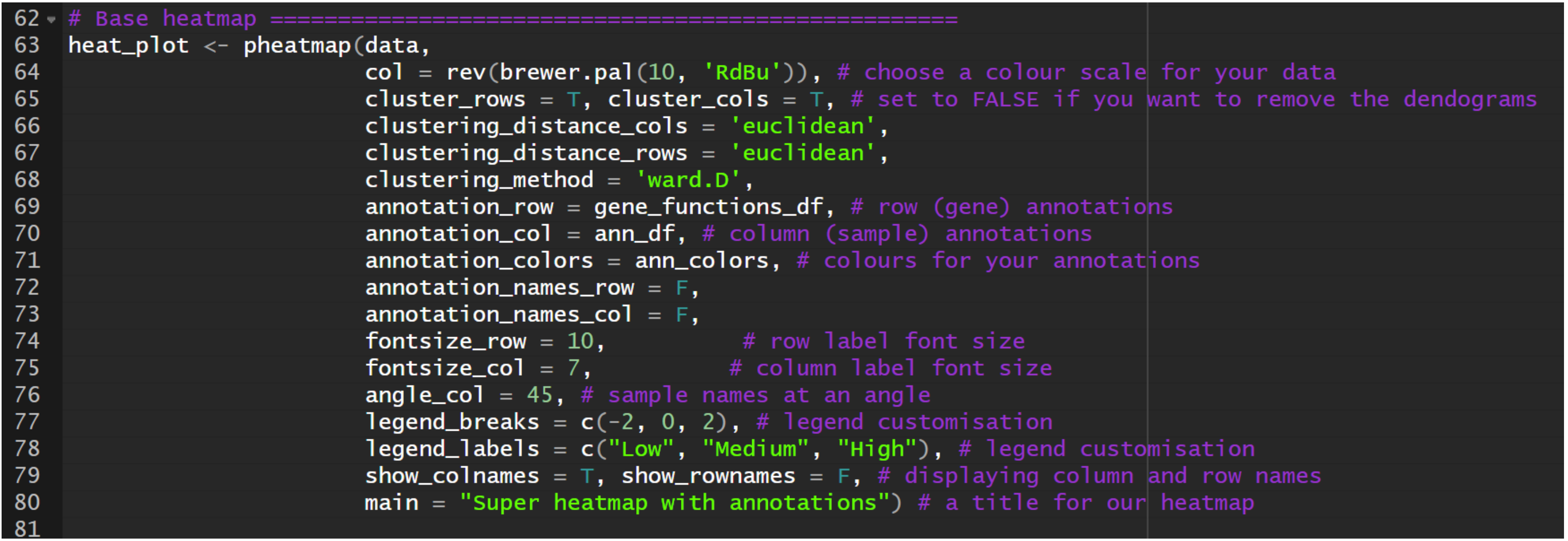
Heatmaps for complex data with pheatmap

- Pheatmap can add clustering of your heatmap rows and columns
- Pheatmap makes additional annotations easier than ggplot

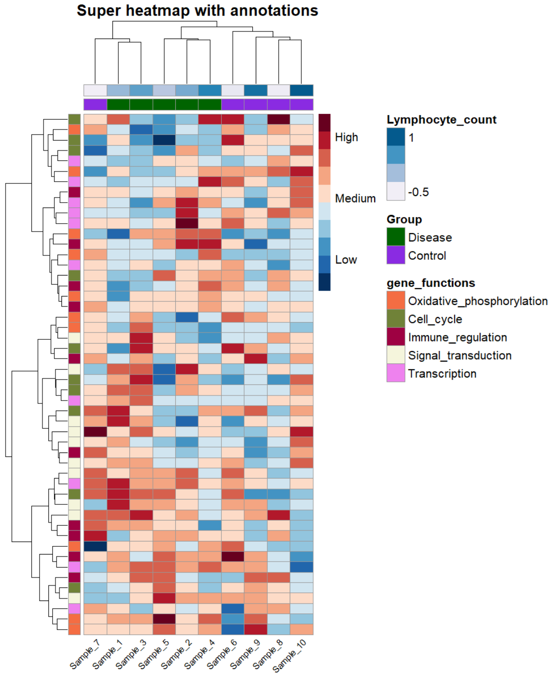


https://biostatsquid.com/step-by-step-heatmap-tutorial-with-pheatmap/

Heatmaps for complex data with pheatmap



```r
# Base heatmap ====================================================
heat_plot <- pheatmap(data,
                      col = rev(brewer.pal(10, 'RdBu')), # choose a colour scale for your data
                      cluster_rows = T, cluster_cols = T, # set to FALSE if you want to remove the dendograms
                      clustering_distance_cols = 'euclidean',
                      clustering_distance_rows = 'euclidean',
                      clustering_method = 'ward.D',
                      annotation_row = gene_functions_df, # row (gene) annotations
                      annotation_col = ann_df, # column (sample) annotations
                      annotation_colors = ann_colors, # colours for your annotations
                      annotation_names_row = F,
                      annotation_names_col = F,
                      fontsize_row = 10,          # row label font size
                      fontsize_col = 7,          # column label font size
                      angle_col = 45, # sample names at an angle
                      legend_breaks = c(-2, 0, 2), # legend customisation
                      legend_labels = c("Low", "Medium", "High"), # legend customisation
                      show_colnames = T, show_rownames = F, # displaying column and row names
                      main = "Super heatmap with annotations") # a title for our heatmap

```



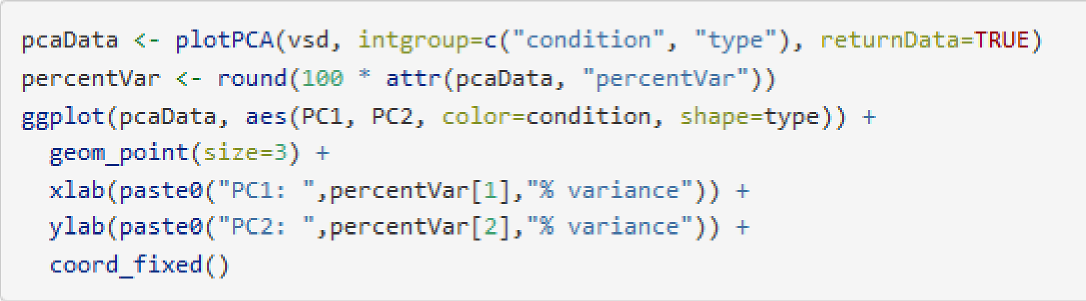
Plots from analysis packages

- DEseq2 plotPCA function

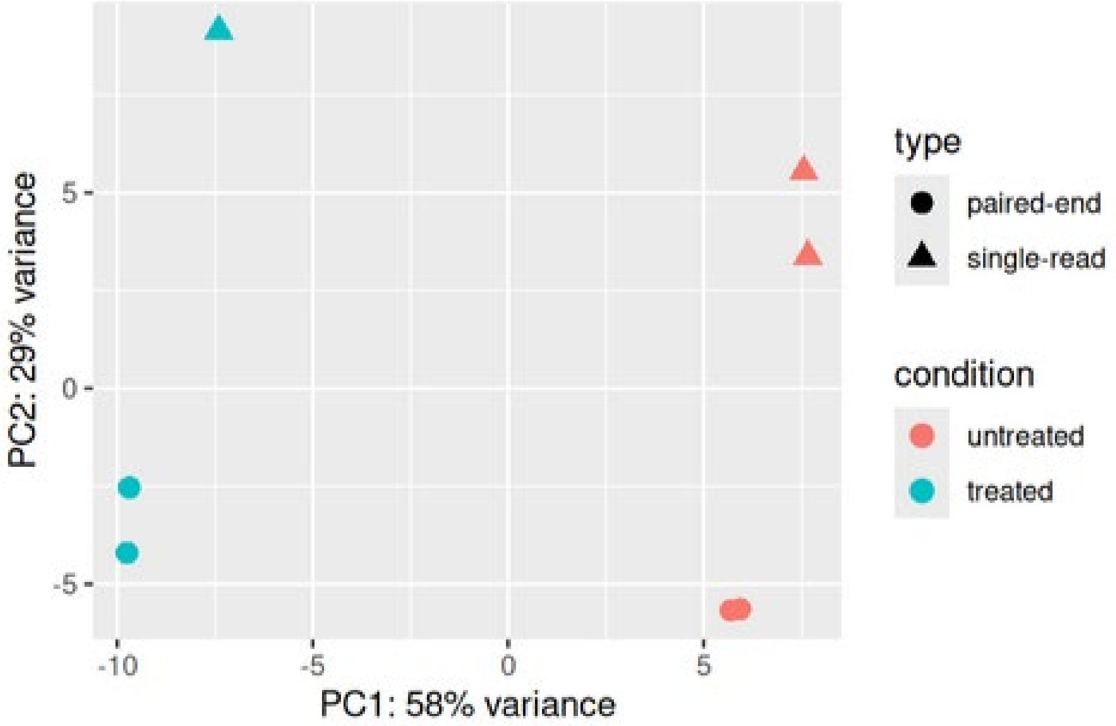
```
plotPCA(vsd, intgroup=c("condition", "type"))
```

https://bioconductor.org/packages/devel/bioc/vignettes/DESeq2/inst/doc/DESeq2.html

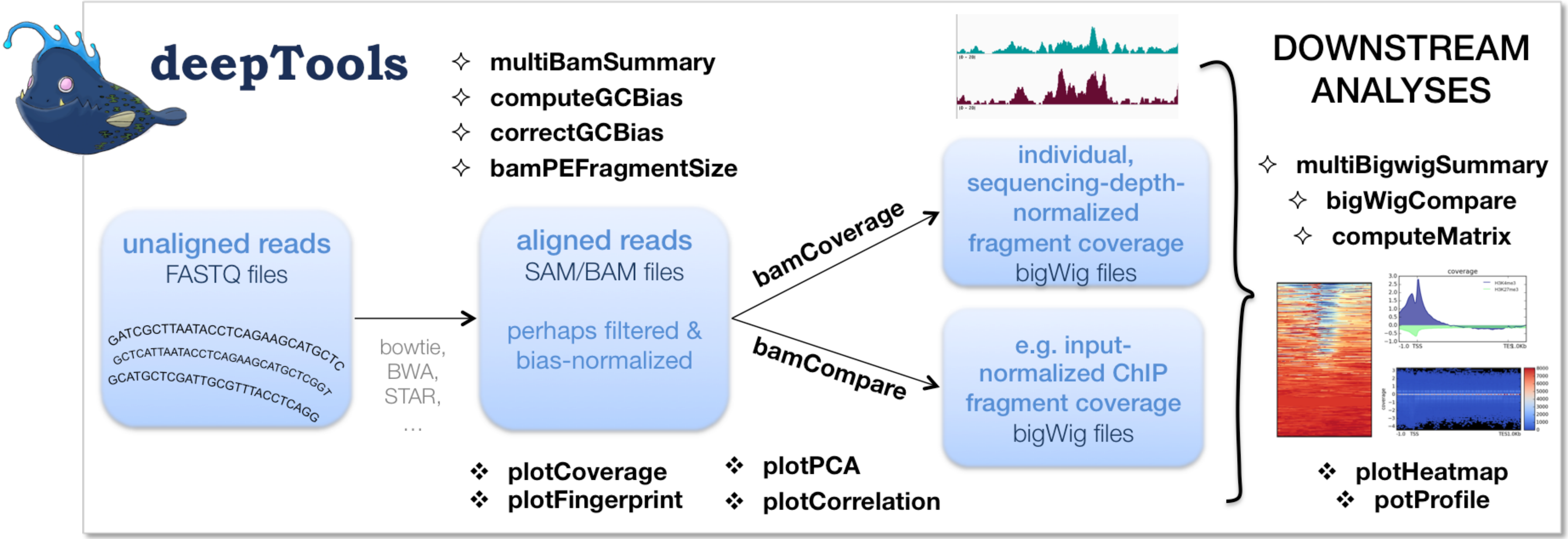
DEseq2 PCA plot is compatible with ggplot



```
pcaData <- plotPCA(vsd, intgroup=c("condition", "type"), returnData=TRUE)
percentVar <- round(100 * attr(pcaData, "percentVar"))
ggplot(pcaData, aes(PC1, PC2, color=condition, shape=type)) +
  geom_point(size=3) +
  xlab(paste0("PC1: ",percentVar[1],"% variance")) +
  ylab(paste0("PC2: ",percentVar[2],"% variance")) +
  coord_fixed()
```



Deeptools for region-based sequencing data



deepTools

- multiBamSummary
- computeGCBias
- correctGCBias
- bamPEFragmentSize

unaligned reads
FASTQ files

GATCGCTTAATACCTCAGAAGCATGCTC
GCTCATTAATACCTCAGAAGCATGCTCGGT
GCATGCTCGATTGCGTTTACCTCAGG

bowtie,
BWA,
STAR,
…

aligned reads
SAM/BAM files

perhaps filtered &
bias-normalized

- plotCoverage
- plotFingerprint
- plotPCA
- plotCorrelation

bamCoverage

bamCompare

individual, sequencing-depth-normalized fragment coverage
bigWig files

e.g. input-normalized ChIP fragment coverage
bigWig files

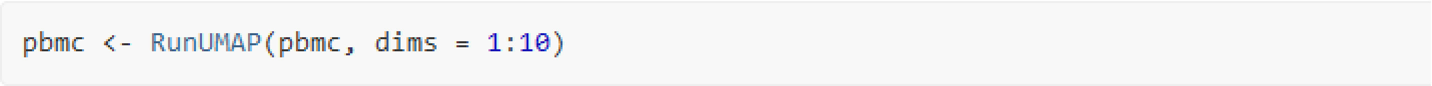
DOWNSTREAM ANALYSES

- multiBigwigSummary
- bigWigCompare
- computeMatrix

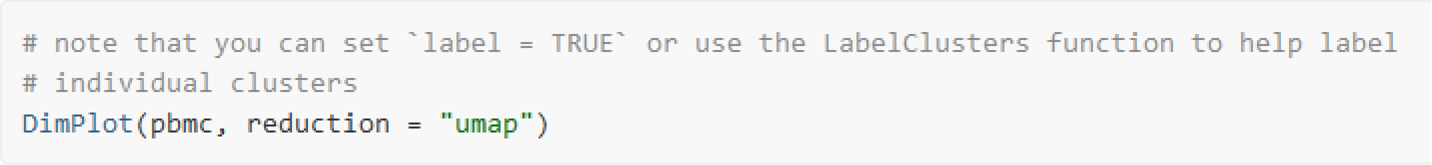
- plotHeatmap
- potProfile

https://deeptools.readthedocs.io/en/develop/

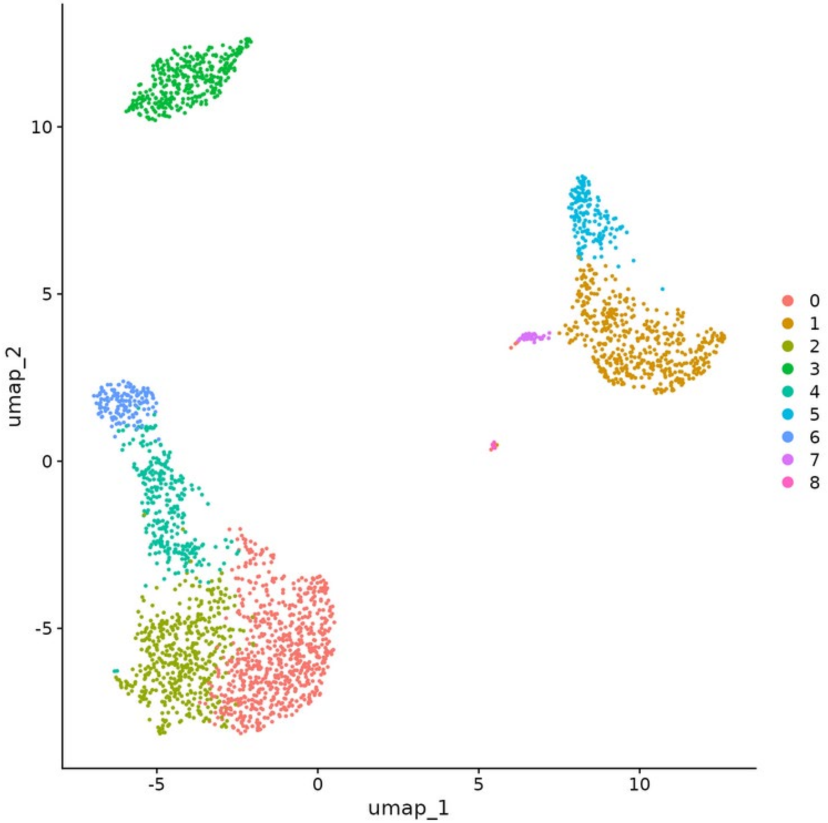
Single cell RNAseq visualizations in Seurat



```
pbmc <- RunUMAP(pbmc, dims = 1:10)
```



```
# note that you can set `label = TRUE` or use the LabelClusters function to help label
# individual clusters
DimPlot(pbmc, reduction = "umap")
```



https://satijalab.org/seurat/articles/pbmc3k_tutorial.html

Shiny apps for interactive visualizations

X variable

Bill Length (mm)

Y variable

Bill Depth (mm)

Filter by species

☑ Adelie

☑ Gentoo

☑ Chinstrap

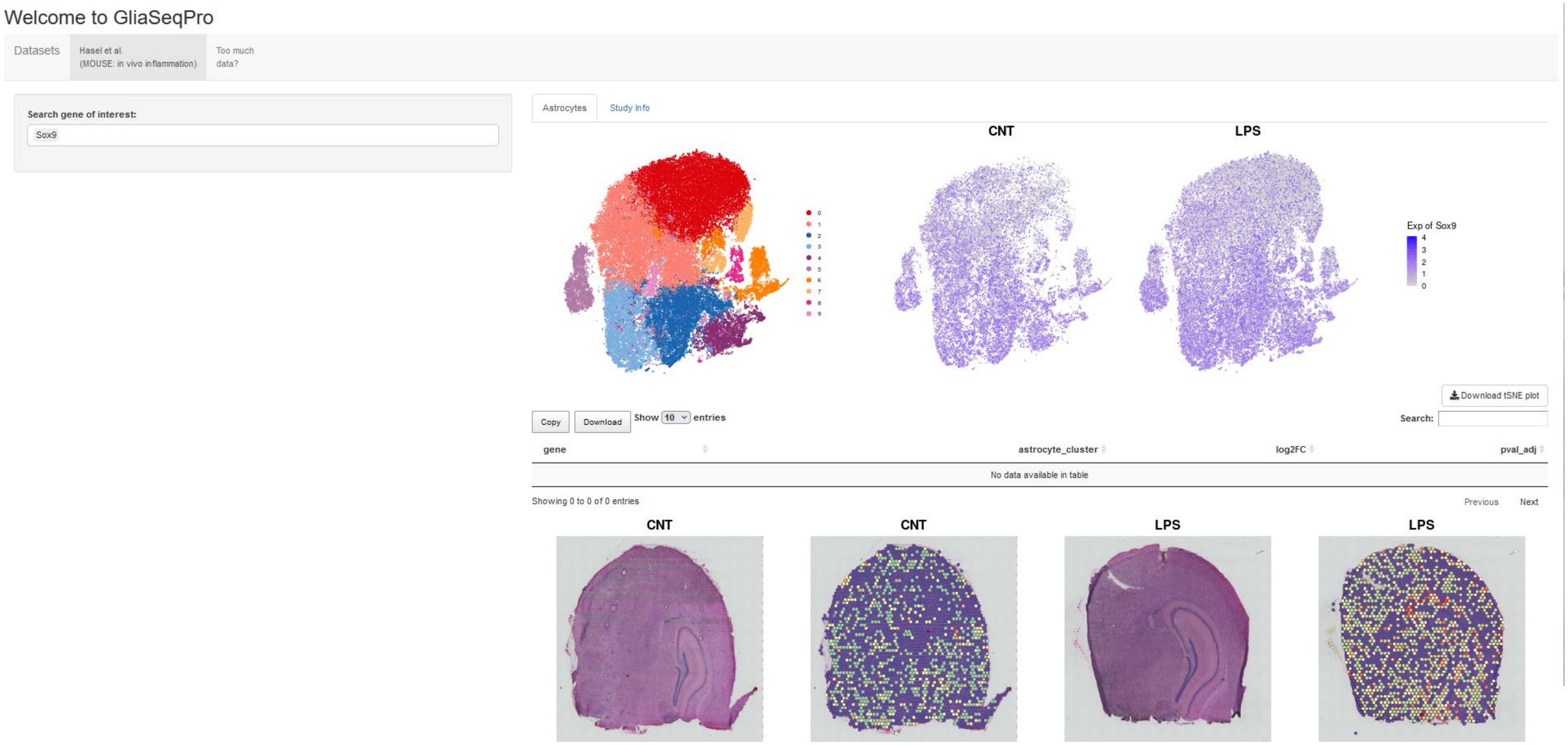
☑ Show species

☑ Show marginal plots

☐ Add smoother

https://shiny.posit.co/

GliaSeqPro Shiny app example



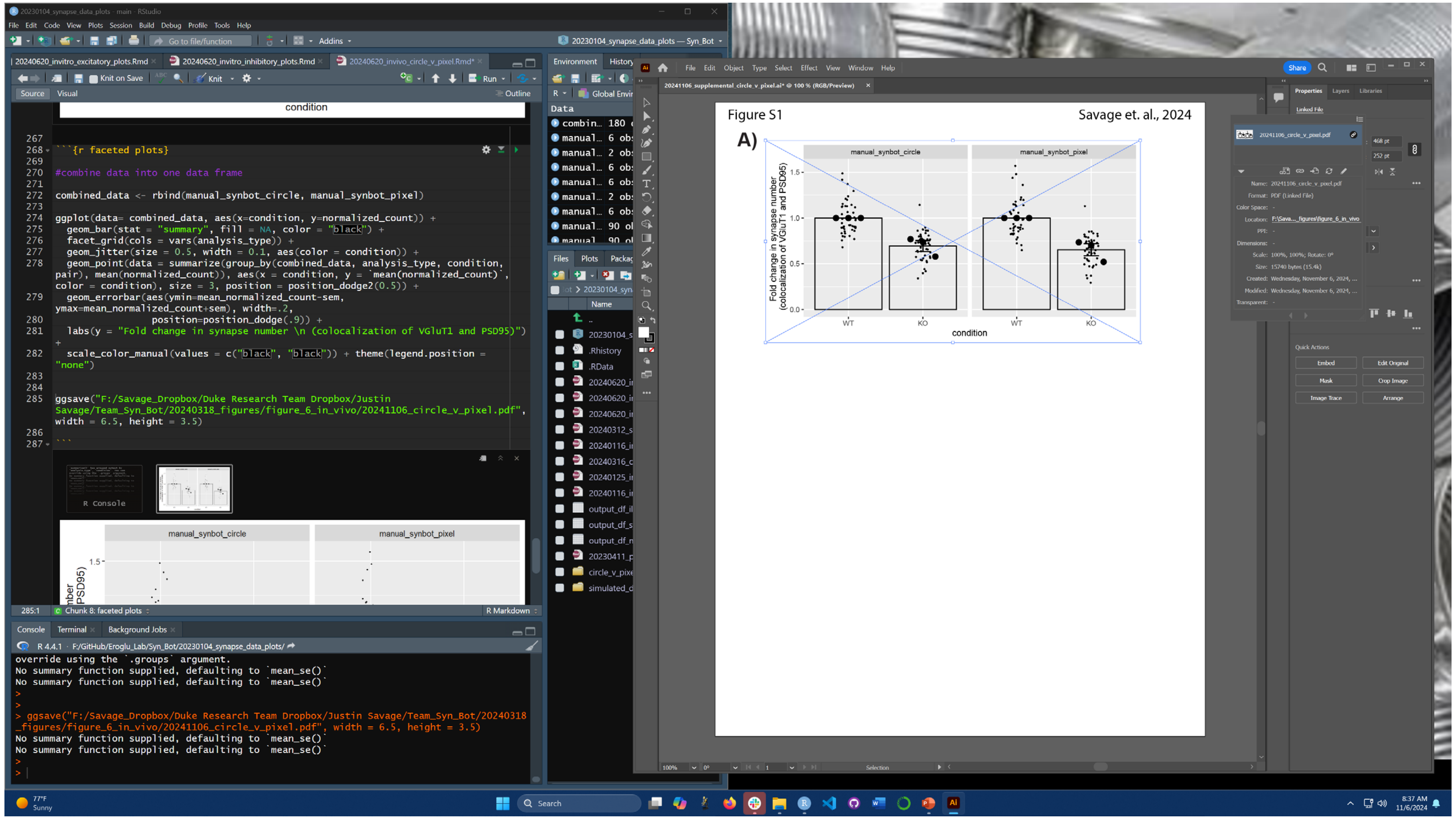
Search gene of interest: Sox9

gene	astrocyte_cluster	log2FC	pval_adj
No data available in table			

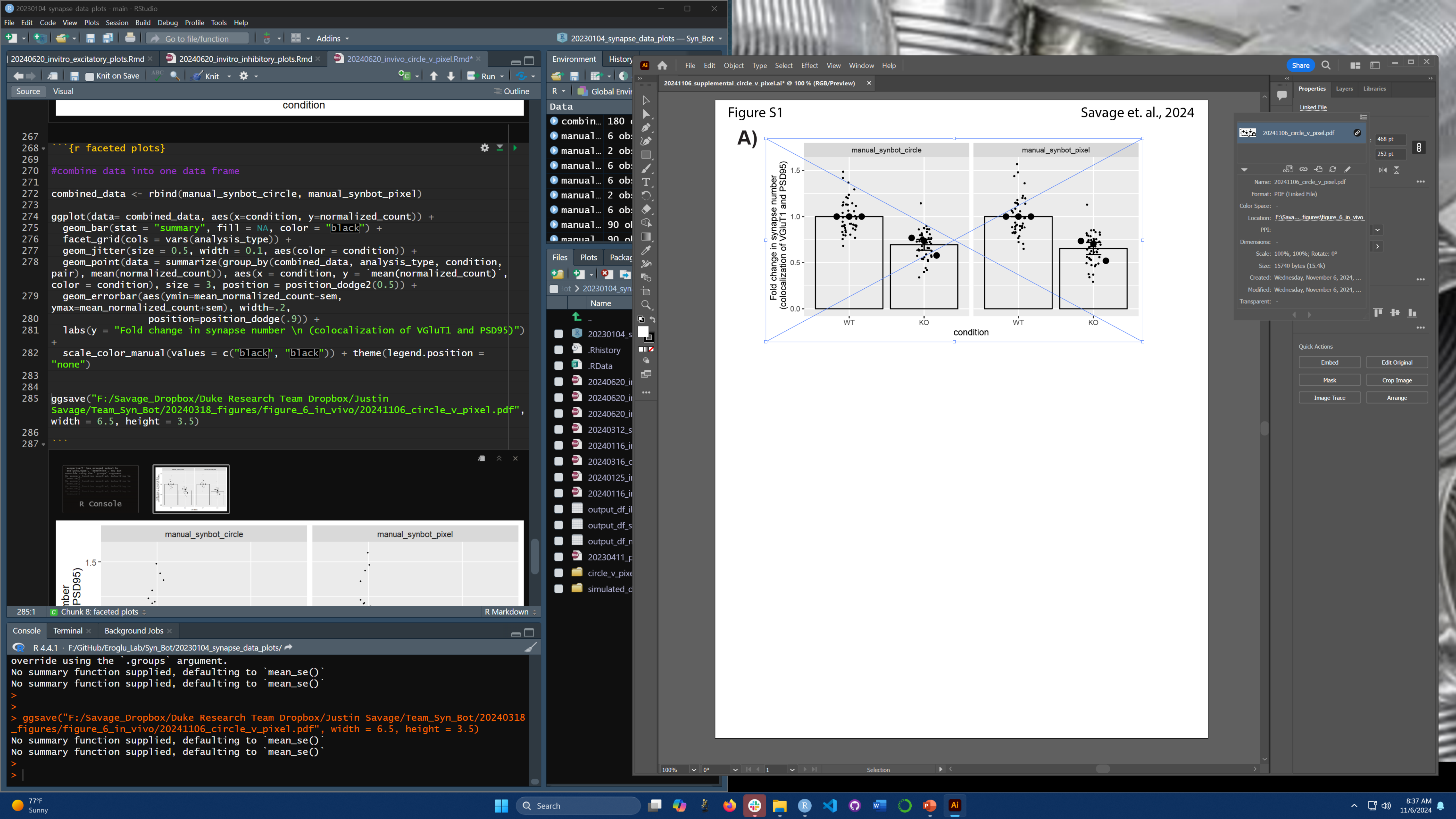
Showing 0 to 0 of 0 entries

https://liddelowlab.shinyapps.io/GliaSeqPro/

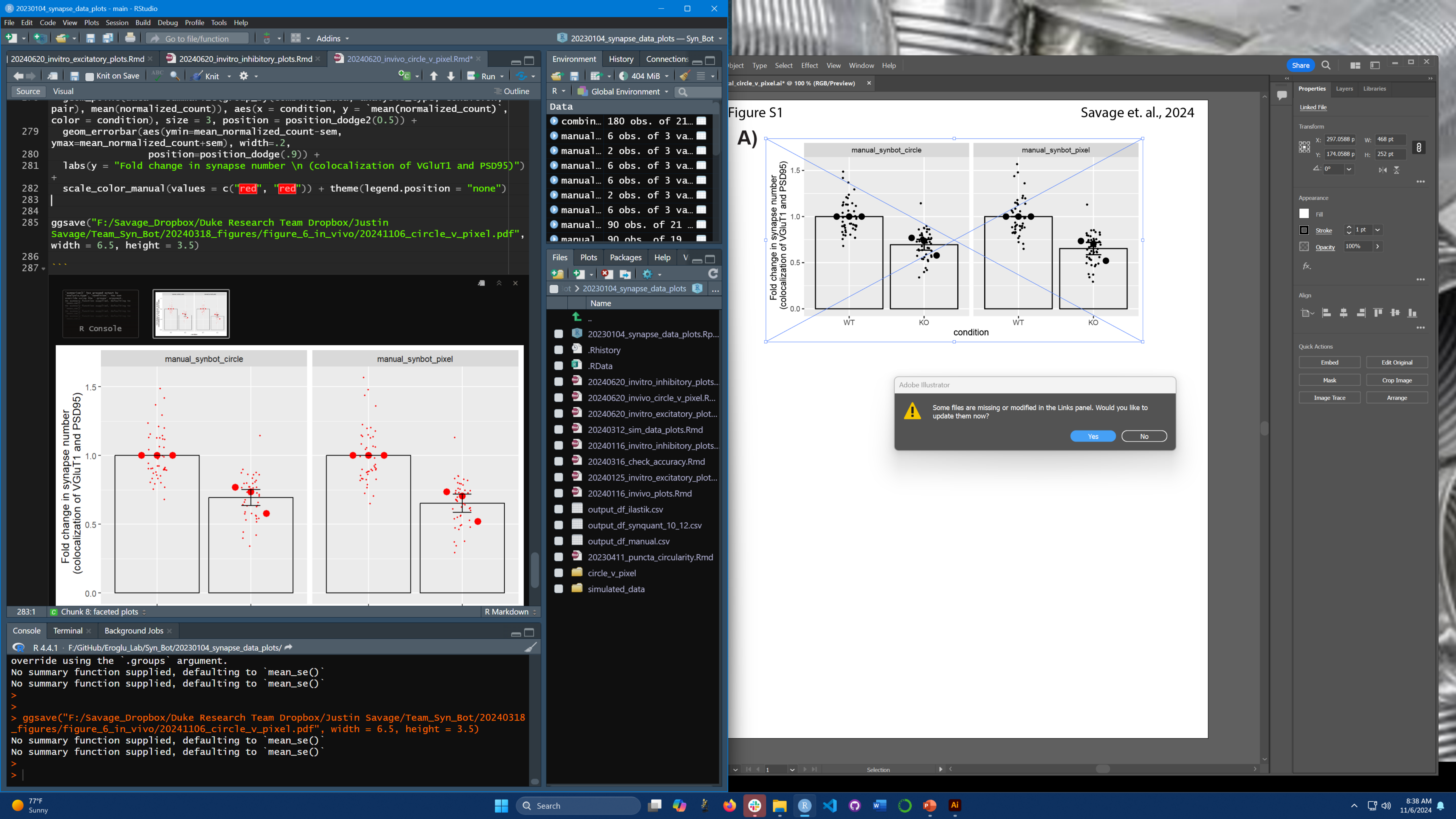
Linking R plots to Adobe Illustrator



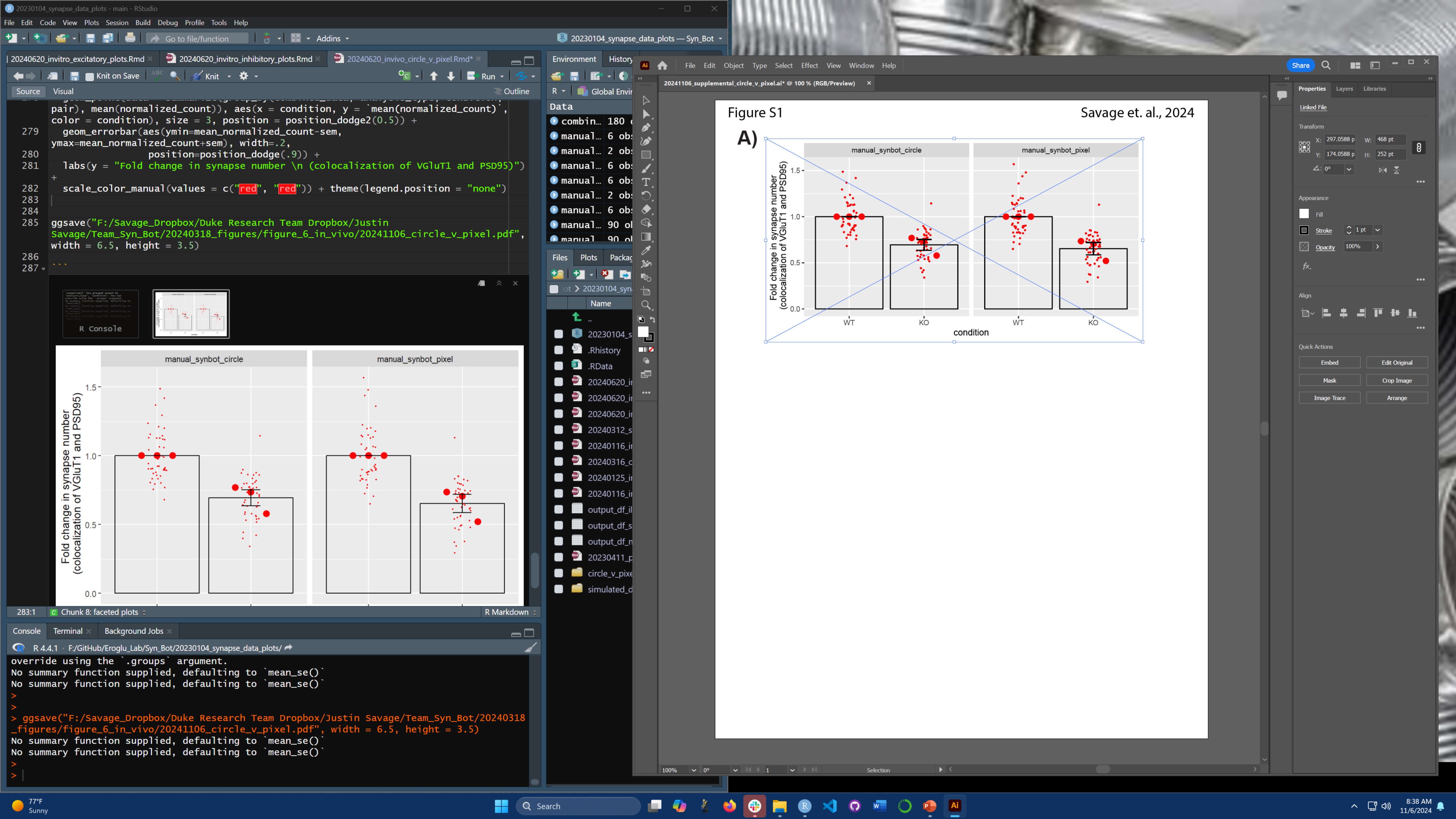
~~~
```{r faceted plots}

#combine data into one data frame

combined_data <- rbind(manual_synbot_circle, manual_synbot_pixel)

ggplot(data= combined_data, aes(x=condition, y=normalized_count)) +
  geom_bar(stat = "summary", fill = NA, color = "black") +
  facet_grid(cols = vars(analysis_type)) +
  geom_jitter(size = 0.5, width = 0.1, aes(color = condition)) +
  geom_point(data = summarize(group_by(combined_data, analysis_type, condition,
pair), mean(normalized_count)), aes(x = condition, y = `mean(normalized_count)`,
color = condition), size = 3, position = position_dodge2(0.5)) +
  geom_errorbar(aes(ymin=mean_normalized_count-sem,
ymax=mean_normalized_count+sem), width=.2,
                 position=position_dodge(.9)) +
  labs(y = "Fold change in synapse number \n (colocalization of VGluT1 and PSD95)")
+
  scale_color_manual(values = c("black", "black")) + theme(legend.position =
"none")


ggsave("F:/Savage_Dropbox/Duke Research Team Dropbox/Justin
Savage/Team_Syn_Bot/20240318_figures/figure_6_in_vivo/20241106_circle_v_pixel.pdf",
width = 6.5, height = 3.5)

~~~
```

```
R 4.4.1 · F:/GitHub/Eroglu_Lab/Syn_Bot/20230104_synapse_data_plots/
override using the `.groups` argument.
No summary function supplied, defaulting to `mean_se()`
No summary function supplied, defaulting to `mean_se()`
>
>
> ggsave("F:/Savage_Dropbox/Duke Research Team Dropbox/Justin Savage/Team_Syn_Bot/20240318
_figures/figure_6_in_vivo/20241106_circle_v_pixel.pdf", width = 6.5, height = 3.5)
No summary function supplied, defaulting to `mean_se()`
No summary function supplied, defaulting to `mean_se()`
>
> 
```

https://helpx.adobe.com/illustrator/using/manage-linked-and-embedded-files.html
```



~~~
```{r faceted plots}

#combine data into one data frame

combined_data <- rbind(manual_synbot_circle, manual_synbot_pixel)

ggplot(data= combined_data, aes(x=condition, y=normalized_count)) +
  geom_bar(stat = "summary", fill = NA, color = "black") +
  facet_grid(cols = vars(analysis_type)) +
  geom_jitter(size = 0.5, width = 0.1, aes(color = condition)) +
  geom_point(data = summarize(group_by(combined_data, analysis_type, condition,
pair), mean(normalized_count)), aes(x = condition, y = `mean(normalized_count)`,
color = condition), size = 3, position = position_dodge2(0.5)) +
  geom_errorbar(aes(ymin=mean_normalized_count-sem,
ymax=mean_normalized_count+sem), width=.2,
                 position=position_dodge(.9)) +
  labs(y = "Fold change in synapse number \n (colocalization of VGluT1 and PSD95)")
+
  scale_color_manual(values = c("black", "black")) + theme(legend.position =
"none")


ggsave("F:/Savage_Dropbox/Duke Research Team Dropbox/Justin
Savage/Team_Syn_Bot/20240318_figures/figure_6_in_vivo/20241106_circle_v_pixel.pdf",
width = 6.5, height = 3.5)

~~~
```

```
override using the `.groups` argument.
No summary function supplied, defaulting to `mean_se()`
No summary function supplied, defaulting to `mean_se()`
>
>
> ggsave("F:/Savage_Dropbox/Duke Research Team Dropbox/Justin Savage/Team_Syn_Bot/20240318
_figures/figure_6_in_vivo/20241106_circle_v_pixel.pdf", width = 6.5, height = 3.5)
No summary function supplied, defaulting to `mean_se()`
No summary function supplied, defaulting to `mean_se()`
>
>
```
```



```
pair), mean(normalized_count)), aes(x = condition, y = `mean(normalized_count)`,
color = condition), size = 3, position = position_dodge2(0.5)) +
  geom_errorbar(aes(ymin=mean_normalized_count-sem,
ymax=mean_normalized_count+sem), width=.2,
                    position=position_dodge(.9)) +
  labs(y = "Fold change in synapse number \n (colocalization of VGluT1 and PSD95)")
+
  scale_color_manual(values = c("red", "red")) + theme(legend.position = "none")

ggsave("F:/Savage_Dropbox/Duke Research Team Dropbox/Justin
Savage/Team_Syn_Bot/20240318_figures/figure_6_in_vivo/20241106_circle_v_pixel.pdf",
width = 6.5, height = 3.5)

```
```

```
override using the `.groups` argument.
No summary function supplied, defaulting to `mean_se()`
No summary function supplied, defaulting to `mean_se()`
>
>
> ggsave("F:/Savage_Dropbox/Duke Research Team Dropbox/Justin Savage/Team_Syn_Bot/20240318
_figures/figure_6_in_vivo/20241106_circle_v_pixel.pdf", width = 6.5, height = 3.5)
No summary function supplied, defaulting to `mean_se()`
No summary function supplied, defaulting to `mean_se()`
>
> 
```

Adobe Illustrator

Some files are missing or modified in the Links panel. Would you like to update them now?
```



```
pair), mean(normalized_count)), aes(x = condition, y = `mean(normalized_count)`,
color = condition), size = 3, position = position_dodge2(0.5)) +
  geom_errorbar(aes(ymin=mean_normalized_count-sem,
ymax=mean_normalized_count+sem), width=.2,
                position=position_dodge(.9)) +
  labs(y = "Fold change in synapse number \n (colocalization of VGluT1 and PSD95)")
+
  scale_color_manual(values = c("red", "red")) + theme(legend.position = "none")

ggsave("F:/Savage_Dropbox/Duke Research Team Dropbox/Justin
Savage/Team_Syn_Bot/20240318_figures/figure_6_in_vivo/20241106_circle_v_pixel.pdf",
width = 6.5, height = 3.5)

```
```

```
override using the `.groups` argument.
No summary function supplied, defaulting to `mean_se()`
No summary function supplied, defaulting to `mean_se()`
>
>
> ggsave("F:/Savage_Dropbox/Duke Research Team Dropbox/Justin Savage/Team_Syn_Bot/20240318
_figures/figure_6_in_vivo/20241106_circle_v_pixel.pdf", width = 6.5, height = 3.5)
No summary function supplied, defaulting to `mean_se()`
No summary function supplied, defaulting to `mean_se()`
>
> 
```
```