Lab 2: Exploring the Visualization of Variation and Track Viewers

[Software needed: web access]

As we discussed in the first lecture, one active area of research in data visualization is how to depict variation / uncertainty in datasets, especially in the case of data that are continuous in nature. In one of the readings (Mason, 2019) in the Additional Resources section of this lab, Tracy Weissgerber, a physiologist at a Mayo Clinic in Minnesota, points out that many scientific journals are now, in fact, requiring that continuous data be displayed using visualizations other than bar charts, such as histograms, box plots, or scatterplots. Our own exploration of some of the Datasaurus Dozen datasets from Matejka and Fitzmaurice (2017) last week showed that, in spite of all x and y values having identical averages and standard deviations (and correlations, for that matter!), they all had very different distributions when plotted on a scatterplot.

In today's lab, we will explore a couple of methods for displaying variation, and then move on to visualizing some genomic data at the nucleotide level with a genome browser tool.

Table 1: Tools covered in this lab (see end of lab for references)

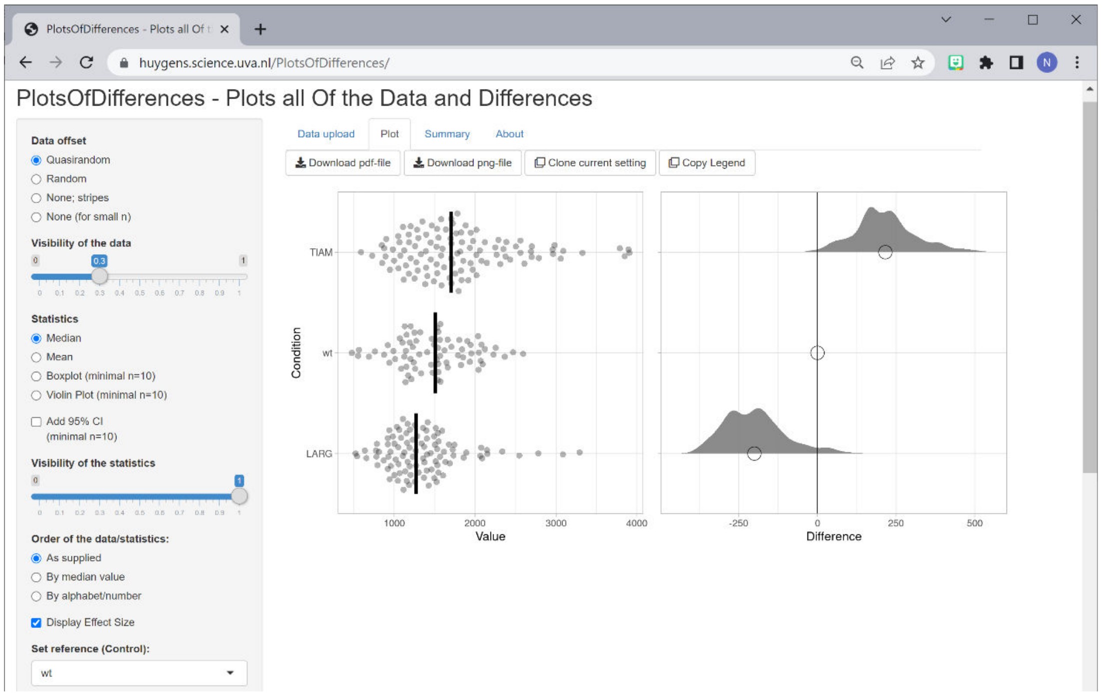
Tool	Notes
PlotsOfDifferences	A Shiny web application (R-based) for generating graphs that displays data in all of their glory. Several customizations are possible. The interface offers several colourblind safe palettes (see this week's lecture material).
R	Violin plots, box plots, and density/rug plots
JBrowse	Exploring RNA-seq data after specifying `min_score` and `max_score` to keep Y axes the same across samples; adding a custom track.

PlotsOfDifferences

1. Go to https://huygens.science.uva.nl/PlotsOfDifferences/. We'll just use the default settings to explore this easy-to-use yet powerful web app.

2. Use the Example 1 dataset in the "Data upload" tab. Switch to the "Plot" tab. You will see two charts as shown in **Figure 1**. Let's assume that the "wt" sample is our control and the TIAM and LARG samples are our "treated" samples that we have repeatedly measured, generating 100+ measurements. Use the "Set reference (Control)" dropdown to change the reference sample to be "wt".

   What do the black vertical bars for the three samples represent? Can you change these to represent something else?


PlotsOfDifferences - Plots all Of the Data and Differences

Data offset
- [✓ Quasirandom]
- Random
- None; stripes
- None (for small n)

Visibility of the data: 0.3

Statistics
- [✓ Median]
- Mean
- Boxplot (minimal n=10)
- Violin Plot (minimal n=10)

☐ Add 95% CI (minimal n=10)

Visibility of the statistics: 1

Order of the data/statistics:
- [✓ As supplied]
- By median value
- By alphabet/number

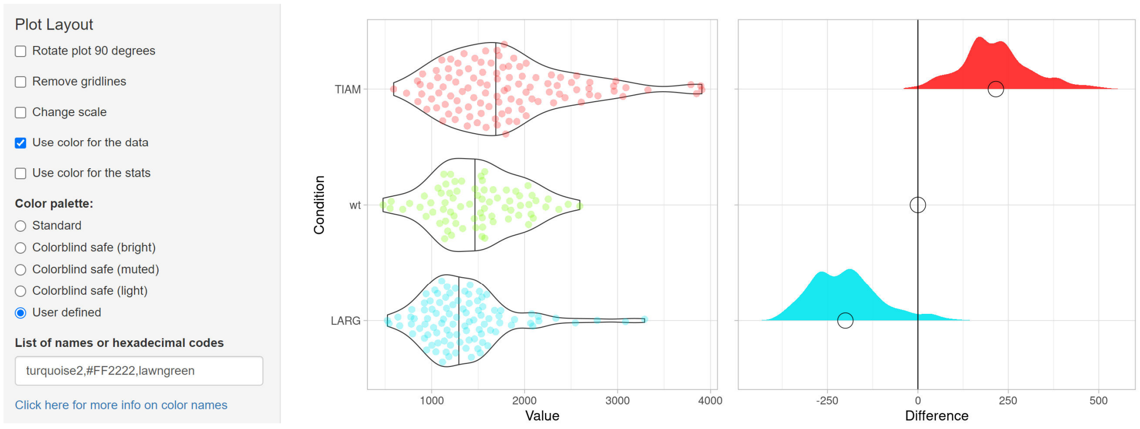
☑ Display Effect Size

Set reference (Control): wt

Figure 1. Examining the Example 1 dataset in PlotOfDifferences, with the reference set to "wt" using the dropdown in the bottom left of the image.

3. The "Quasirandom" data offset option provides a nice layout for each data point in the three datasets. If you have a small number of points, "None" might be suitable, and the "None; stripes" option provides a "rug" effect, which we'll explore with ggplot too in the next section.

4. Try out the Boxplot and Violin Plot options, and add some colour to your charts using the options on the left panel as shown in **Figure 2**.


Plot Layout
- ☐ Rotate plot 90 degrees
- ☐ Remove gridlines
- ☐ Change scale
- ☑ Use color for the data
- ☐ Use color for the stats

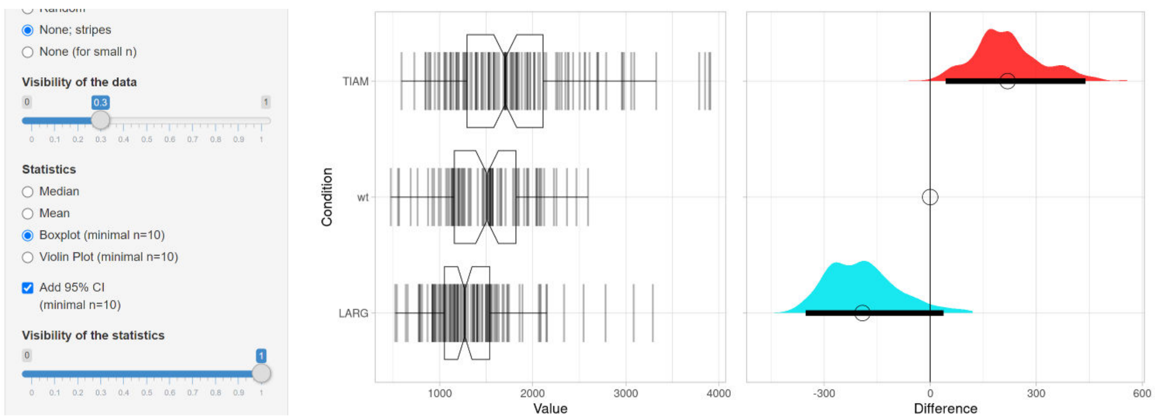
Color palette:
- Standard
- Colorblind safe (bright)
- Colorblind safe (muted)
- Colorblind safe (light)
- [✓ User defined]

List of names or hexadecimal codes: turquoise2,#FF2222,lawngreen

Figure 2. The Violin Plot Statistics option with user defined colours for the colour palette.

5. Change the "Data offset" option to "None; stripes". This will display stripes instead of data points. The density of data for a given x value is reflected by the number of stripes at that point.

6. The plots shown on the right plot are of the differences between central tendency (mean or median) and the samples. The 95% compatibility interval of the difference per condition is calculated from 1000 bootstraps, whose distribution is shown in the difference plot generated using the statistic, which is the mean or median of the reference (i.e. control) sample.



Random
[✓ None; stripes]
None (for small n)

Visibility of the data: 0.3

Statistics
Median
Mean
[✓ Boxplot (minimal n=10)]
Violin Plot (minimal n=10)

[✓ Add 95% CI (minimal n=10)]

Visibility of the statistics: 1

Figure 3. Showing stripes for data points, instead of dots. This creates a bar-code like plot. The 95% compatibility interval is shown in the differences plot. Box plots have been overlaid on the data points.

Lab Quiz Question 1
*Answer lab quiz questions while doing lab!

Look at the p-values by activating the "Display p-value" option in the "Summary" tab. Are these differences significant? The author of PlotsOfDifferences writes of the samples "both [TIAM and LARG] indicate that the observed difference is unlikely given the hypothesis that the samples (condition and perturbation) were acquired from the same population. For TIAM the p-value is compatible with the effect size, while the p-value for LARG seems on the low side, given that the CI of the effect size includes 0. <u>This stresses the careful evaluation of the outcome of statistical tests in relation to the actual data that is acquired.</u>"

PlotOfDifferences provides a quick and easy tool for generating graphs that show the underlying variation in the data. But what about if we want more control over our plotting options?

<u>R</u>

We will explore generating distribution plots, box plots, and violin plots with R. In this case, we will use a new dataset that encompasses Covid-19 cases and deaths, and hospitalizations from Covid-19, in the province of Ontario for the period from March 2020, until October 2022. While this is not a "genomic" dataset *per se*, it is quite large and has a number of categories, so provides a good test case for looking at variation! Download this dataset from the course website. It is called "Ontario_age_and_sex_COVID-19_data_merged_filtered.csv". It was retrieved and slightly reformatted from https://www.publichealthontario.ca/en/data-and-analysis/infectious-disease/covid-19-data-surveillance/covid-19-data-tool?tab=ageSex.

1. Upload the "Ontario_age_and_sex_COVID-19_data_merged_filtered.csv" to your Jupyter notebook as per Step 3 of the R part of Lab 1. Again, we've provided an empty notebook where you could paste code, or an R Notebook with all the code and the file.

2. As for last week's lab, we'll load some packages that will help us with our goal of creating some plots this week.

```
# Packages to help tidy our data
library (tidyverse)
# Packages for the graphical analysis section
library (repr)
# packages used for working with/formatting dates in R
library (lubridate)
library (zoo)
```

   You will receive several messages back about these packages having been attached successfully.

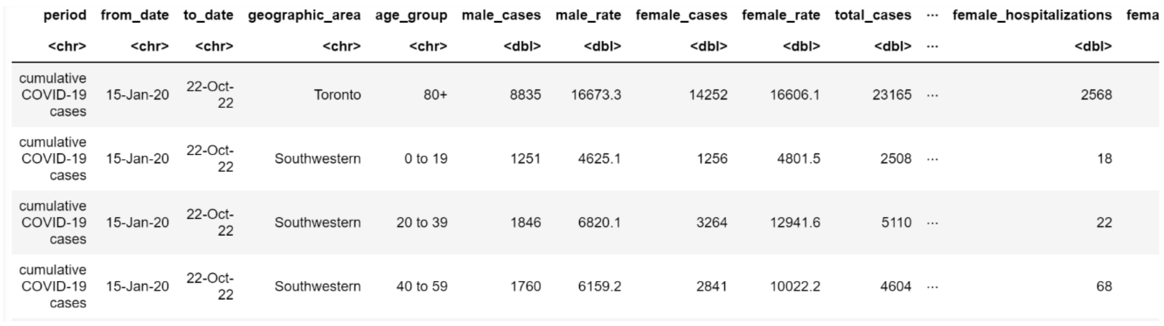
3. Load the dataset from your working directory, which should be `'/home/jovyan/work'` if you're using Coursera's Jupyter Hub.

```
covid_demographics.df <-
read_csv("Ontario_age_and_sex_COVID-19_data_merged.csv")
str(covid_demographics.df)
tail(covid_demographics.df)
```

```
Rows: 204 Columns: 23
── Column specification ────────────────────────────────────────
Delimiter: ","
chr   (5): Period, From date, To date, Geographic area, Age group
dbl  (18): Male Cases, Male Rate, Female Cases, Female Rate, Total Cases, Tot...

spec_tbl_df [204 × 23] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ Period                         : chr [1:204] "cumulative COVID-19 cases" "cumulative...
 $ From date                      : chr [1:204] "15-Jan-20" "15-Jan-20" "15-Jan-20"...
```

4. From the **`tail()`** command above, you will see a snippet of the data frame we have created, as shown in **Figure 4**. Try using **`head(covid_demographics.df, n=50)`** to examine the data, which is always good practice. *How many geographic areas can you see with this command? How many age groups?*



period	from_date	to_date	geographic_area	age_group	male_cases	male_rate	female_cases	female_rate	total_cases	⋯	female_hospitalizations	fema
<chr>	<chr>	<chr>	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	⋯	<dbl>	
cumulative COVID-19 cases	15-Jan-20	22-Oct-22	Toronto	80+	8835	16673.3	14252	16606.1	23165	⋯	2568	
cumulative COVID-19 cases	15-Jan-20	22-Oct-22	Southwestern	0 to 19	1251	4625.1	1256	4801.5	2508	⋯	18	
cumulative COVID-19 cases	15-Jan-20	22-Oct-22	Southwestern	20 to 39	1846	6820.1	3264	12941.6	5110	⋯	22	
cumulative COVID-19 cases	15-Jan-20	22-Oct-22	Southwestern	40 to 59	1760	6159.2	2841	10022.2	4604	⋯	68	

Figure 4. A snippet of our "`covid_demographics.df`" data frame.

5. We are going to first use some "regular expressions" to replace some characters in our data frame with other characters, for instance, we'll replace spaces with underscores in the column names. We'll also use a similar "regex" mechanism to filter our dataset. The chain operator (`%>%`) allows the output of one operation to feed into the next.

```
# Format our dataset to focus on total_cases and total_deaths
covid_demographics_total.df <-
covid_demographics.df %>%
# Pare down the dataset to just total_cases, total_deaths, and
# total_hospitalizations
select(from_date, to_date, geographic_area, age_group,
total_cases, total_deaths, total_hospitalizations_count) %>%
# Convert the age_group into a "factor"
mutate(age_group = factor(age_group)) %>%
rename(public_health_unit = geographic_area) %>%
# Group the data so you can "summarize" in the next steps
group_by(public_health_unit) %>%
# Generate percent cases for each age group within a PHU
mutate(percent_cases = total_cases/sum(total_cases),
# Generate percent deaths for each age group within a PHU
percent_deaths = total_deaths/sum(total_deaths),
# Generate % hospitalizations for each age group within a PHU
percent_hospitalizations =
total_hospitalizations_count/sum(total_hospitalizations_count))
# Take a look at the different age demographics
levels(covid_demographics_total.df$age_group)
```

```
'0 to 19''20 to 39''40 to 59''60 to 79''80+'
```

6. Now we can plot what the distribution of our data looks like, based on the age bands indicated above, from each of Ontario's 34 public health units (PHUs).

```
# Adjust our plot window size according to the expected output
options(repr.plot.width=20, repr.plot.height=10)
covid_demographics_total.df %>%

# Filter for some of our age groups
filter(age_group %in% c("20 to 39","40 to 59", "60 to 79",
"80+")) %>%

# 1. Data
ggplot(.) +

# 2. Aesthetics
aes(x = percent_cases, fill = age_group) +
theme(text = element_text(size = 20)) + # set text size

# 4. Geoms
geom_density(alpha = 0.5) + # generate kernel density estimate
geom_rug(aes(colour = age_group))
# confirm our data values with a geom_rug
```

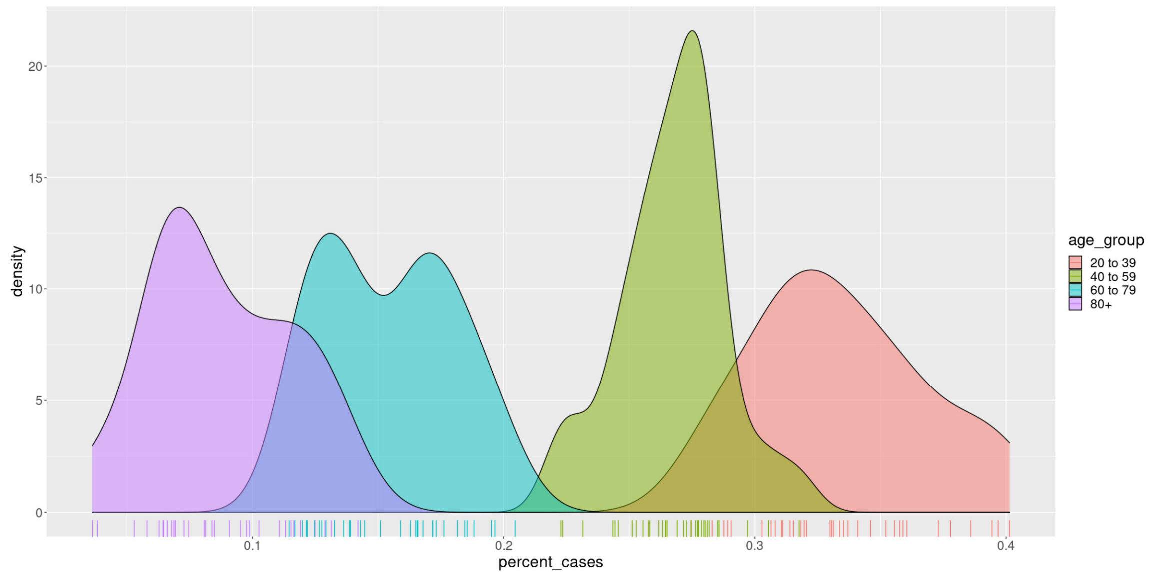


Figure 5. Density of the percent of Covid-19 cases in 4 age groups.

The plot in **Figure 5** shows us how abundant the percentage of cases in 4 age bands is, based on data from 34 public health units in Ontario. Each public health unit datum for each age band is denoted with a little stripe in the "rug" below the plot.

Which age group had the most cases during the Covid-19 pandemic?

Lab Quiz Question 2

7. If we have more than 3 or 4 categories, we might want to use the powerful **`facet_wrap`** function of ggplot. This creates small multiples of graphs, with each dataset plotted separately on its own miniature graph. One important data visualization aspect to keep in mind is to use the same scale for each graph. This way, a viewer only has to understand what the x and y coordinates are once and then at a glance can easily see where the maxima and minima are for a given dataset. This might not always be useful, though…

```
# Instantiate ggplot object
# Adjust our plot window size according to the expected output

options(repr.plot.width=20, repr.plot.height=20)
covid_demographics_total.df %>%

# Select for just the important columns

select(public_health_unit, age_group, percent_cases,
percent_deaths) %>%

# 1. Data
ggplot(.) +
# 2. Aesthetics
aes(x = percent_cases, fill = age_group) +
theme (text = element_text(size = 20)) + # set text size

# 4. Geoms
geom_density(alpha = 0.5) +

# 6. Facet
facet_wrap(~age_group, ncol = 1 )
# try with scales = "free_y" as a parameter
```

Try re-generating the plot shown in **Figure 6** by adding the **`scales = "free_y"`** parameter to the **`facet_wrap`** function.

What happens to the y axis scales when you do this? Does this make more sense?

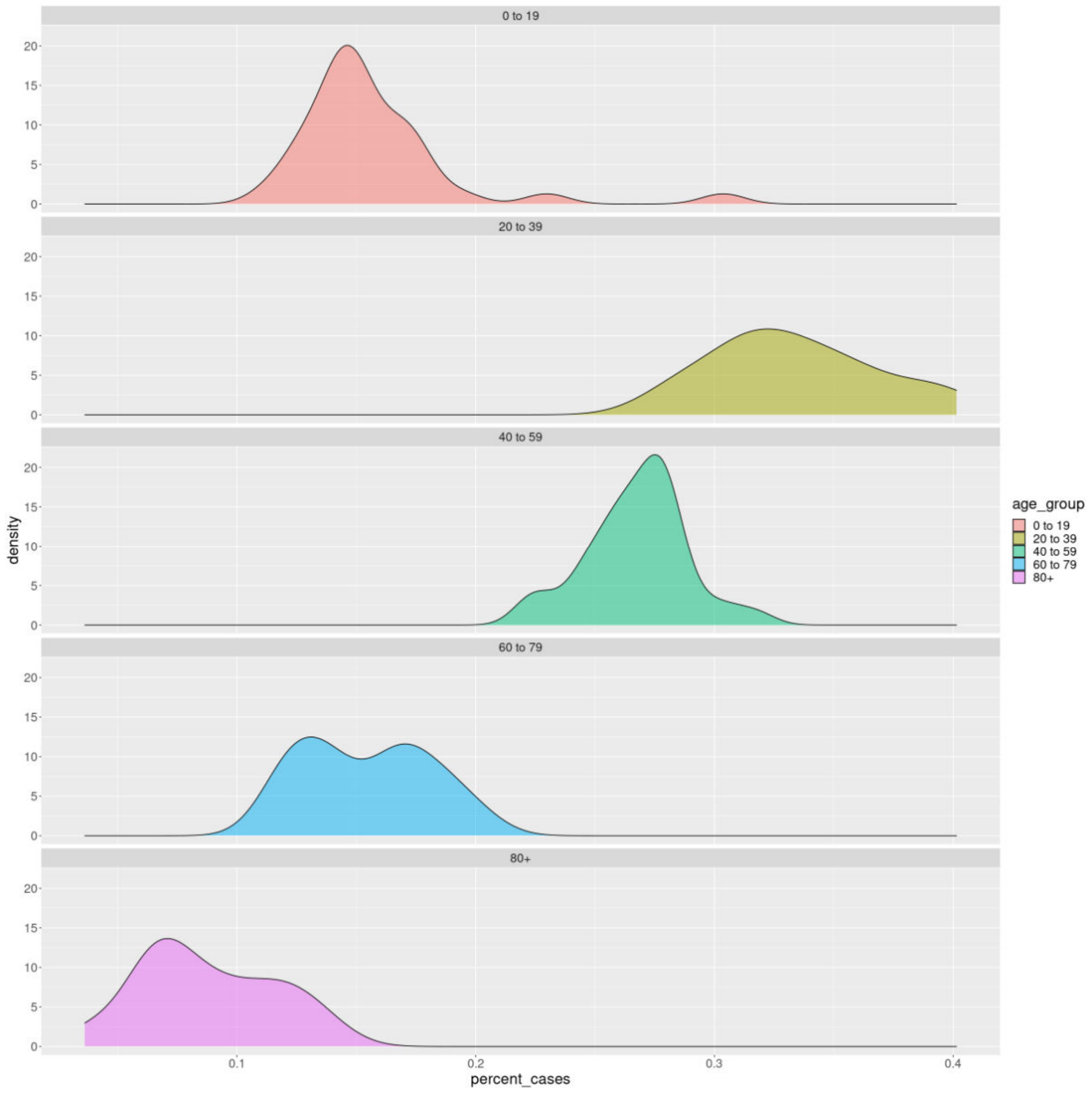


Figure 6. Creating small multiples of graphs with the **`facet_wrap`** function in ggplot.

8. Next, we'll explore plotting both Covid-19 cases and deaths in the same figure using boxplots, along with showing all data points, added as jitter plots overlaid on the boxplots. See **Figure 7**.

```
# Adjust our plot window size according to the expected output
options(repr.plot.width=20, repr.plot.height=10)
covid_demographics_total.df %>%

# Select for just the important columns

select(public_health_unit, age_group, percent_cases,
percent_deaths) %>%

# Pivot the modified table to capture the "stat_group" of
# percent_cases vs percent_deaths

pivot_longer(cols=c(3:4), names_to = "stat_group", values_to =
"percent_PHU_total") %>%

# Plot the data as a grouped boxplot

# 1. Data
ggplot(.) +

# 2. Aesthetics
aes(x=age_group, y = percent_PHU_total, fill = stat_group) +
theme(text = element_text(size = 20)) + # set text size
theme(legend.position = "right") +
scale_y_continuous(limits = c(0, 0.6)) + # Set the y-axis limit

# 4. Geoms
geom_boxplot(outlier.shape=NA, notch = FALSE) +
# Add the boxplot geom

geom_point(position=position_jitterdodge(jitter.width = 0.1,
jitter.height = 0, dodge.width = 0.75, seed = NA), alpha =
0.25) # Add data points for comparison
```

The **`jitter.width`** parameter controls the dispersion of the data points within each box plot. Try adjusting this parameter and other parameters to see what aspect they control!

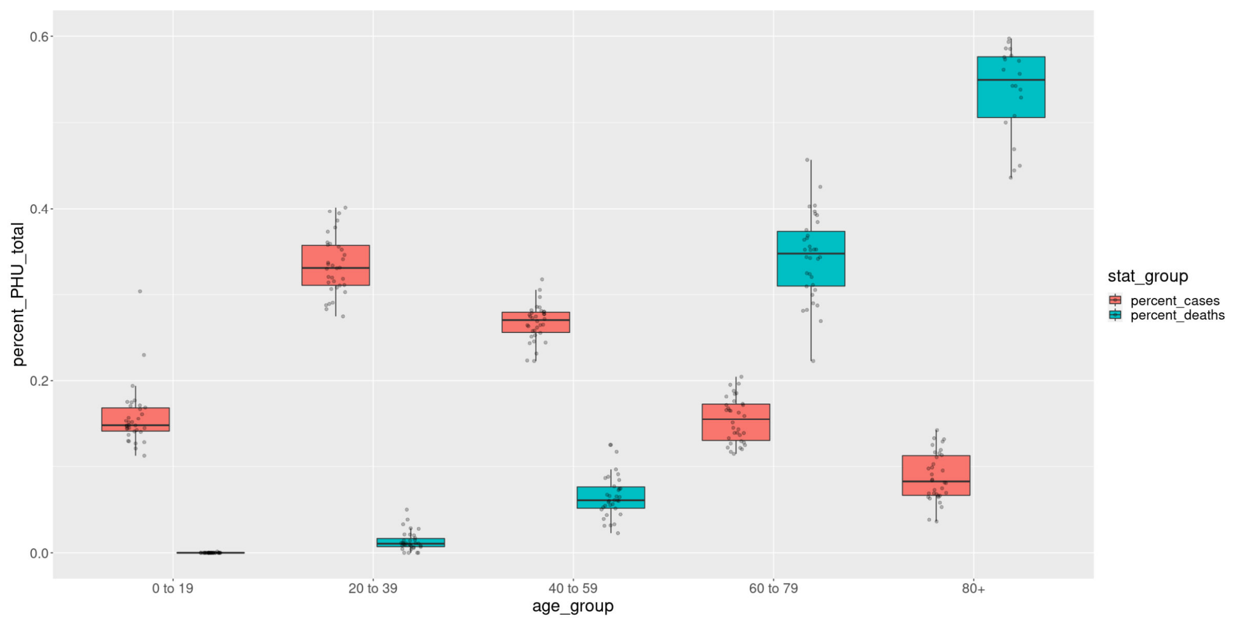


Figure 7. Covid-19 cases and deaths plotted by age group. Each dot on a box plot represents a datum from a single public health unit. The "tightness" of the jittered points is controlled with the `jitter.width` parameter.

Which age bracket was most susceptible to dying from Covid-19?

9. Box plots can represent the summary information of the distribution of data in a dataset but are always a visual representation of an even distribution. There are not enough parameters supplied to represent anything more complex! Violin plots are not limited in that respect. Despite some of their visual caveats, violin plots can help in detecting multi-modal data. It is recommended to use violin plots when you have at least 30 data points.

```
# Adjust our plot window size according to the expected output
options(repr.plot.width=20, repr.plot.height=10)
# Generate a basic box plot with outliers present
covid_demographics_total.df %>%
# 1. Data
ggplot(.) +
# 2. Aesthetics
aes(x=age_group, y = percent_cases) +
theme(text = element_text(size = 20)) + # set text size
theme(legend.position = "none") +
# 4. Geoms
geom_violin() + # Add the boxplot geom
geom_jitter(aes(colour = public_health_unit), width = 0.1,
height = 0)
```

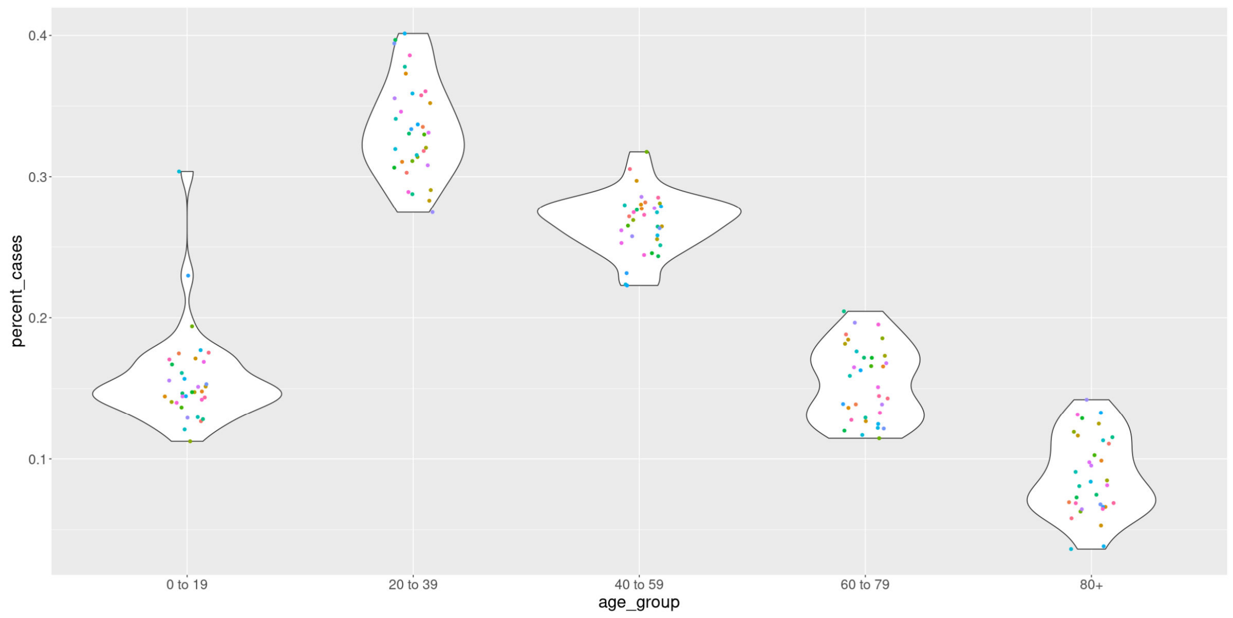


Figure 8. Violin plots of the number of cases by age group. As we saw in Figures 5 and 6, the data from individual public health units do not exhibit normal distributions across age groups.

JBrowse

JBrowse is a powerful JavaScript application for exploring genome-scale data at the nucleotide level. You can think of many of the nucleotide-resolution tracks as tens of thousands of bar graphs. It is possible to upload your own data fairly easily to a JBrowse instance to compare your own data with tracks that are available in public JBrowse instances.

We'll just briefly examine how to do one crucial thing in JBrowse, namely, how to set the Y axes of transcript mapping plots such that all Y axes use the same scale. In some way these tracks are like small multiples and thus comparison of expression levels in different tissues is facilitated by having the same scale for all tracks.

1. Go to the link in the footnote below[1]. Make sure all information in this URL is present in the browser location bar. This will open up three RNA-seq tracks from our favourite reference plant species, *Arabidopsis thaliana*. You should see something that looks like **Figure 9**.

[1] https://jbrowse.arabidopsis.org/index.html?data=Araport11&loc=Chr2%3A15203387..15207961&tracks=TAIR10_genome%2Cflower12_tophat%2Cleaf_tophat%2Croot_tophat%2CA11-PC-Oct22&highlight=

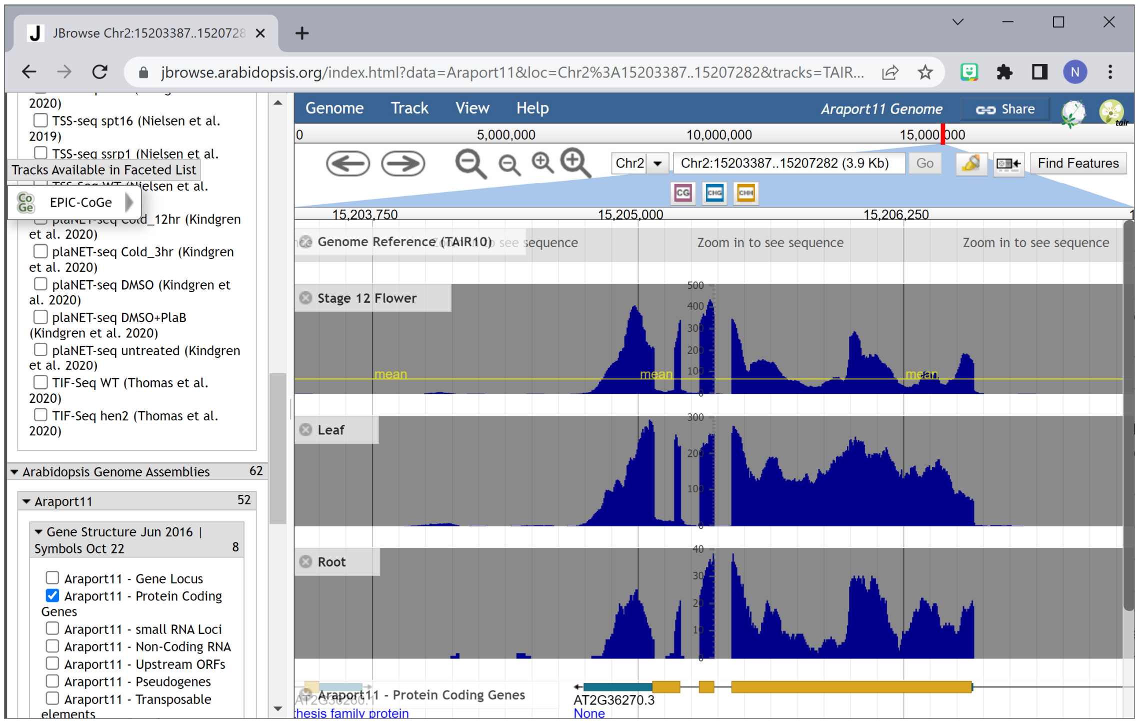

JBrowse Chr2:15203387..1520728
jbrowse.arabidopsis.org/index.html?data=Araport11&loc=Chr2%3A15203387..15207282&tracks=TAIR...
Tracks Available in Faceted List
EPIC-CoGe
TSS-seq spt16 (Nielsen et al. 2019)
TSS-seq ssrp1 (Nielsen et al.
plaNET-seq Cold_12hr (Kindgren et al. 2020)
plaNET-seq Cold_3hr (Kindgren et al. 2020)
plaNET-seq DMSO (Kindgren et al. 2020)
plaNET-seq DMSO+PlaB (Kindgren et al. 2020)
plaNET-seq untreated (Kindgren et al. 2020)
TIF-Seq WT (Thomas et al. 2020)
TIF-Seq hen2 (Thomas et al. 2020)
Arabidopsis Genome Assemblies 62
Araport11 52
Gene Structure Jun 2016 | Symbols Oct 22 8
Araport11 - Gene Locus
Araport11 - Protein Coding Genes
Araport11 - small RNA Loci
Araport11 - Non-Coding RNA
Araport11 - Upstream ORFs
Araport11 - Pseudogenes
Araport11 - Transposable elements
Genome
Track
View
Help
Araport11 Genome
Share
Chr2
Chr2:15203387..15207282 (3.9 Kb)
Go
Find Features
Genome Reference (TAIR10)
Zoom in to see sequence
Stage 12 Flower
Leaf
Root
mean
Araport11 - Protein Coding Genes
AT2G36270.3
None

Figure 9: TAIR's JBrowse instance showing RNA-seq mapping coverage for RNA-seq data from 3 different tissues. Which tissue exhibits the lowest level of expression of At2g36270?

2. For the three RNA-seq based evidence tracks for Stage 12 Flower, Leaf, and Root, click on the small down arrow ▾ that will appear when you hover over the track name. Click on the Edit config option and then replace `"autoscale": "local"` with `"min_score": 0,` followed by `"max_score": 500,` on the next line. Click Apply. You can also add a bit of colour by specifying `"pos_color": "orange",` or another named colour. Do this for all three tissues. See **Figure 10** for this output.

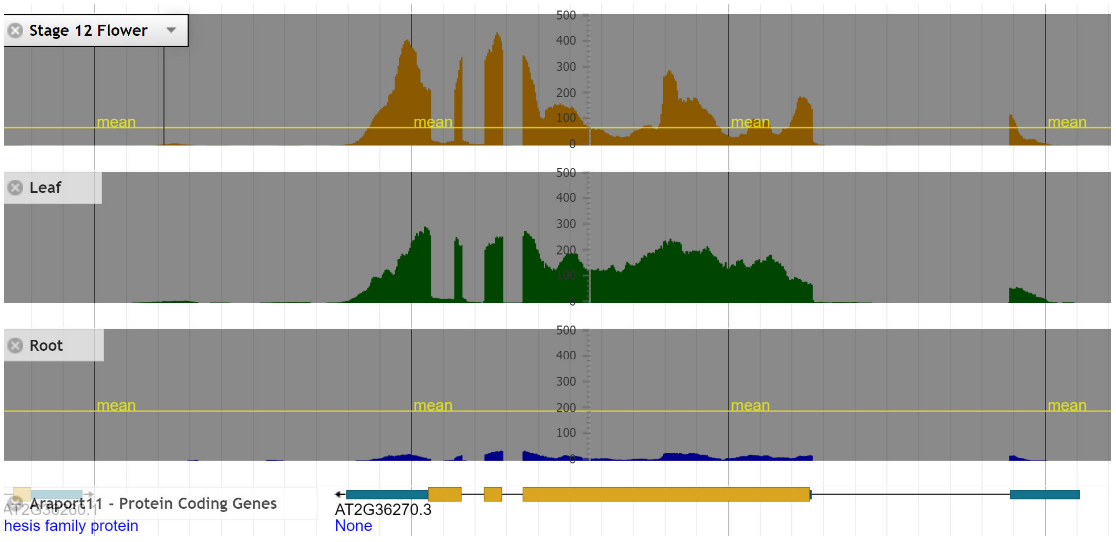


Figure 10. RNA-seq mapping coverage with Y axes scales set to have the same maximum and minimum values.

In which sample is the level of gene expression lowest for At2g36270?

In some cases, having the Y axes be autoscaled makes sense, such as when you're examining the data for potential intron retention events. And for statistically calling differentially expressed genes, programs like edgeR and DESeq2 are typically used (see Week 5 of Bioinformatic Methods II for greater detail). While scaling the y-axes to be the same in this exercise seems like a good idea, we might want to consider how many reads were actually mapped (see the Lab Discussion video for a bit more information). Check out other so-called "wiggle" track configuration options at http://gmod.org/wiki/JBrowse_Configuration_Guide#Wiggle_track_configuration_options. There are many track viewers available, including IGB, IGV and Circos, for generating circular plots. They are well worth exploring…but remember your data visualization best practices when using these.

3. Last, with a small amount of effort, we can also display our own tracks in any JBrowse instance, too. Let's say we've developed an algorithm that will call a genomic region as being an exon or not based on RNA-seq data (note, there are many such algorithms, this is just a made-up example!). One of the basic formats accepted by JBrowse and other track viewers is Wiggle format. This is simply a text file with the first line describing some parameters, then second line the format and chromosome, and the remaining lines containing data on a genomic coordinate-by-genomic coordinate basis, like so:

```
Track type=wiggle_0 name="variableStep" description="variableStep
format" visibility=full autoScale=on viewLimits=0.0:100.0
color=50,150,255 yLineMark=11.76 yLineOnOff=on priority=10
variableStep chrom=chr2
15203748 1
15203749 1
15203750 1
15203751 1
…
15204072 100
15204073 100
15204074 100
15204075 100
…
```

In this file we've given a score for a "non-exon" of 1, and for an "exon" to 100 based on where we see RNA-seq data mapping in Figures 9 and 10. After converting this file to BigWig format (a binary, indexed version of the file) using Galaxy as shown in the Appendix, we can upload it to TAIR's JBrowse instance and, provided we are in the genomic position as shown in Figures 9 and10, we will see our "Exon Predictor" values, as shown in **Figure 11**! This is just a trivial example, but it shows the power of JBrowse for displaying both publicly-available tracks together with data we generate ourselves.

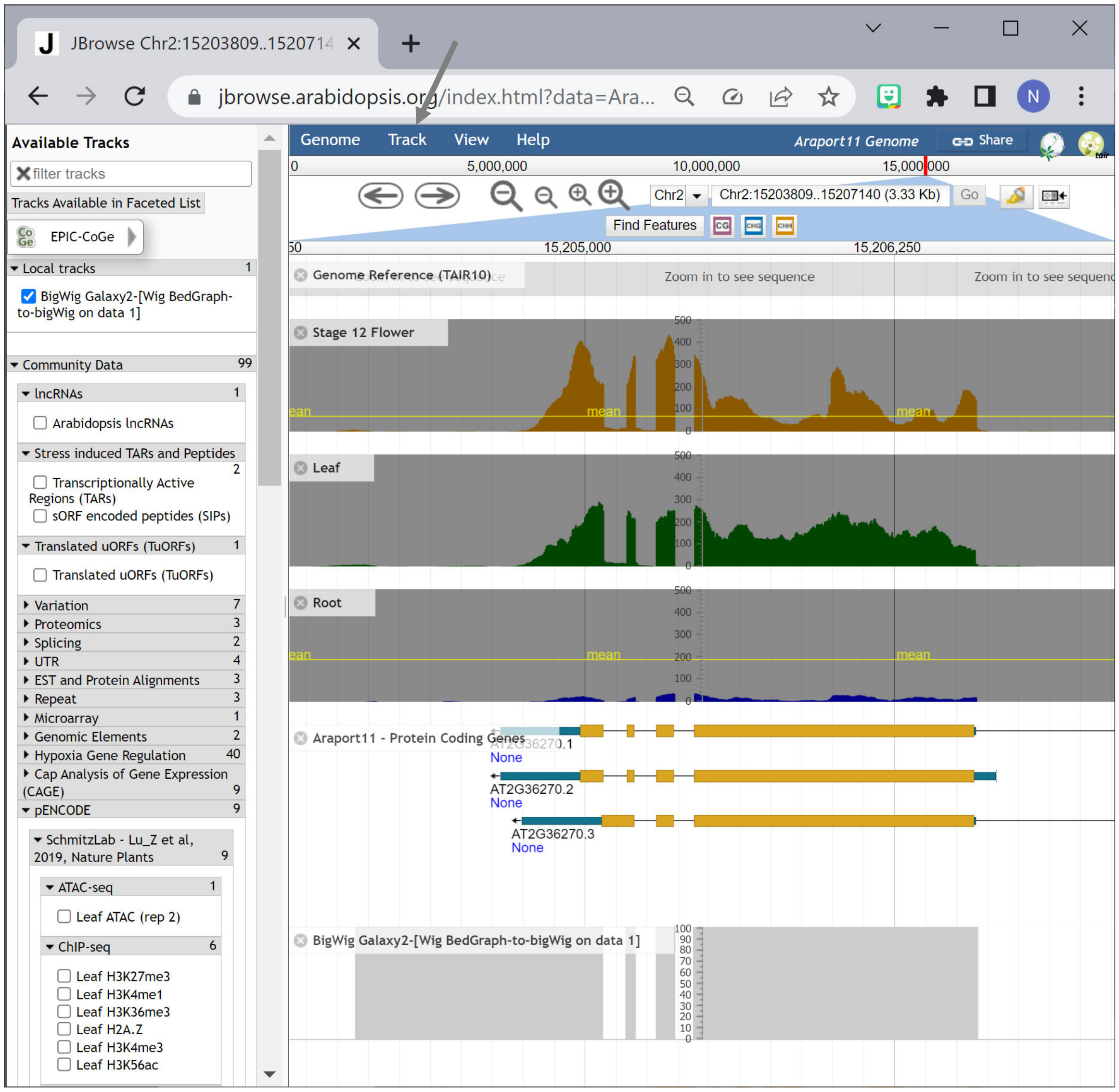

JBrowse Chr2:15203809..152071
jbrowse.arabidopsis.org/index.html?data=Ara...
Available Tracks
filter tracks
Tracks Available in Faceted List
EPIC-CoGe
Local tracks 1
BigWig Galaxy2-[Wig BedGraph-to-bigWig on data 1]
Community Data 99
lncRNAs 1
Arabidopsis lncRNAs
Stress induced TARs and Peptides 2
Transcriptionally Active Regions (TARs)
sORF encoded peptides (SIPs)
Translated uORFs (TuORFs) 1
Translated uORFs (TuORFs)
Variation 7
Proteomics 3
Splicing 2
UTR 4
EST and Protein Alignments 3
Repeat 3
Microarray 1
Genomic Elements 2
Hypoxia Gene Regulation 40
Cap Analysis of Gene Expression (CAGE) 9
pENCODE 9
SchmitzLab - Lu_Z et al, 2019, Nature Plants 9
ATAC-seq 1
Leaf ATAC (rep 2)
ChIP-seq 6
Leaf H3K27me3
Leaf H3K4me1
Leaf H3K36me3
Leaf H2A.Z
Leaf H3K4me3
Leaf H3K56ac
Genome
Track
View
Help
Araport11 Genome
Share
Chr2
Chr2:15203809..15207140 (3.33 Kb)
Go
Find Features
Genome Reference (TAIR10)
Zoom in to see sequence
Stage 12 Flower
Leaf
Root
mean
Araport11 - Protein Coding Genes
AT2G36270.1
AT2G36270.2
AT2G36270.3
None
BigWig Galaxy2-[Wig BedGraph-to-bigWig on data 1]

Figure 11: Adding our own BigWig track to a public JBrowse instance. Use the Track function along the top to upload our BigWig file.

End of Lab!

Lab 2 Objectives

By the end of Lab 2 (comprising the lab including its boxes, and the lecture), you should:

- understand the Gestalt principles of human visual perception;
- appreciate that context biases perception;
- be aware that our colour vision is limited;
- be able to create simple plots with PlotOfDifferences;
- be able to generate distributions, box plots, and violin plots with R;
- be cognizant of the importance of using the same scales for small multiple graphs, including those in JBrowse.

Do not hesitate to check with the Forums for this course on Coursera if you do not understand any of the above after reading the relevant material.

Additional Resources

Justin Matejka and George Fitzmaurice. 2017. Same Stats, Different Graphs: Generating Datasets with Varied Appearance and Identical Statistics through Simulated Annealing. Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems. Association for Computing Machinery, New York, NY, USA, 1290-1294. DOI: https://doi.org/10.1145/3025453.3025912.

Joachim Goedhart (2019). PlotsOfDifferences - a web app for the quantitative comparison of unpaired data. BioRXiv. https://doi.org/10.1101/578575

Also check out the Goedhart Lab's PlotsOfData: https://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.3000202 and SuperPlotsOfData https://www.molbiolcell.org/doi/pdf/10.1091/mbc.E20-09-0583 !

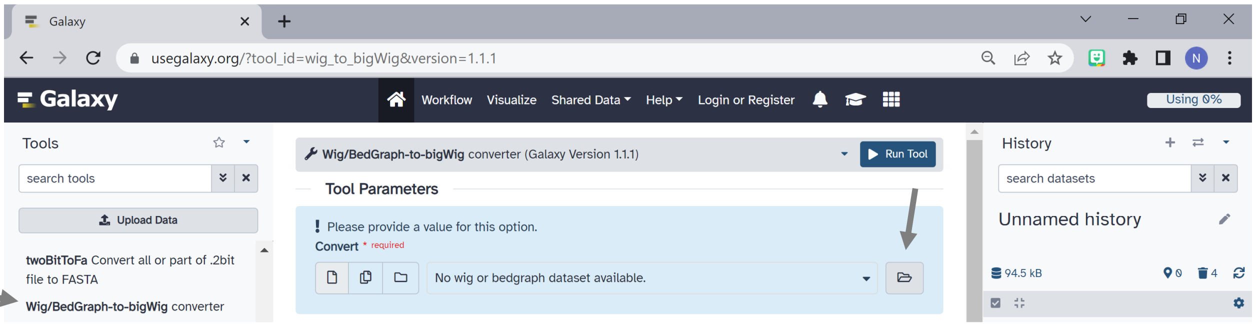
Yan Holtz, https://r-graph-gallery.com/index.html (code is available to help create any of these graphs!)

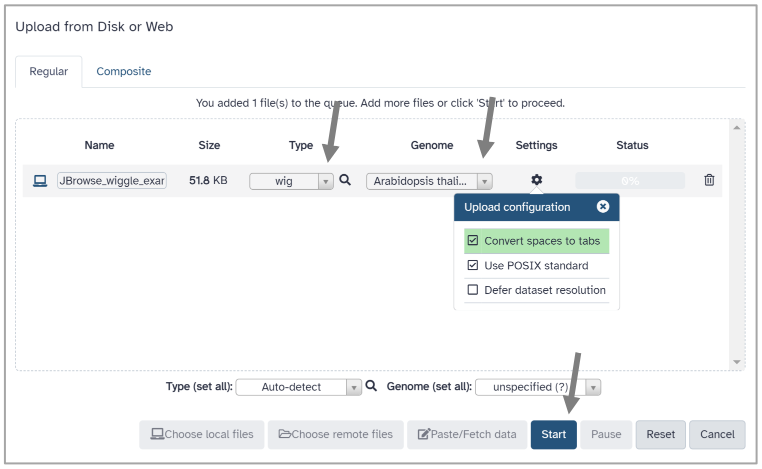
Betsy Mason (2019) Why scientists need to be better at data visualization . Knowable Magazine. Online at https://knowablemagazine.org/article/mind/2019/science-data-visualization

http://gmod.org/wiki/JBrowse_Configuration_Guide#Wiggle_track_configuration_options

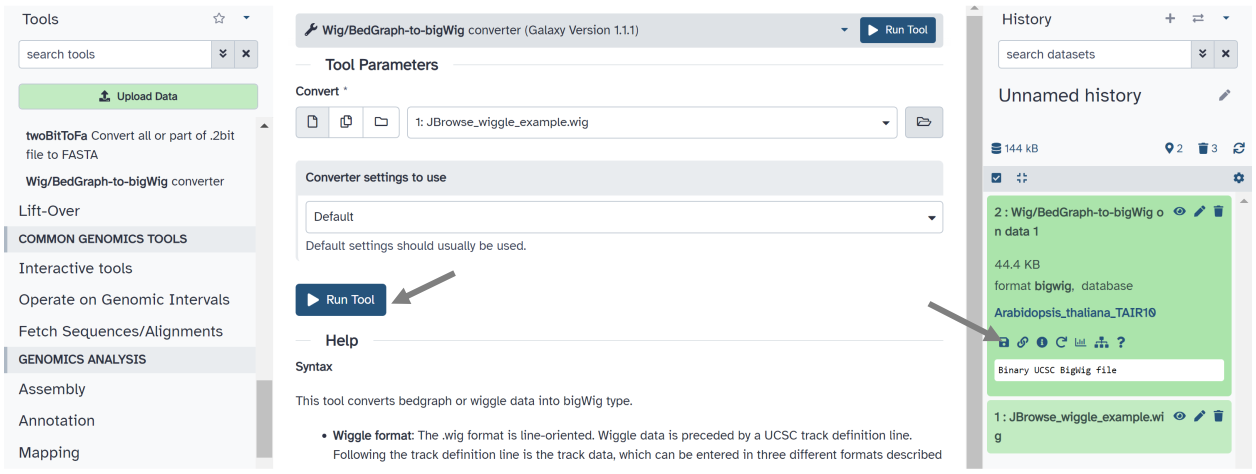
Appendix: Converting a text-based Wiggle file to a binary, indexed BigWig file with Galaxy

Go to usegalaxy.org and choose the **Wig/BedGraph-to-BigWig** converter in the left panel, under GENOMIC FILE MANIPULATION, in the Convert Formats subsection. Upload the provided Wiggle file using the folder icon in the middle panel with the following settings, **Type** = wig and **Genome** = Arabidopsis thaliana TAIR10 and click Start (cancel out of the Upload box to return to the main screen):


Galaxy
usegalaxy.org/?tool_id=wig_to_bigWig&version=1.1.1
Workflow
Visualize
Shared Data
Help
Login or Register
Using 0%
Tools
search tools
Upload Data
twoBitToFa Convert all or part of .2bit file to FASTA
Wig/BedGraph-to-bigWig converter
Wig/BedGraph-to-bigWig converter (Galaxy Version 1.1.1)
Run Tool
Tool Parameters
Please provide a value for this option.
Convert * required
No wig or bedgraph dataset available.
History
search datasets
Unnamed history
94.5 kB
0
4



After clicking ▶Run Tool and waiting a while, download the Bigwig file on the right by clicking on the disk icon, to use in JBrowse.


Tools
search tools
Upload Data
twoBitToFa Convert all or part of .2bit file to FASTA
Wig/BedGraph-to-bigWig converter
Lift-Over
COMMON GENOMICS TOOLS
Interactive tools
Operate on Genomic Intervals
Fetch Sequences/Alignments
GENOMICS ANALYSIS
Assembly
Annotation
Mapping
Wig/BedGraph-to-bigWig converter (Galaxy Version 1.1.1)
Run Tool
Tool Parameters
Convert *
1: JBrowse_wiggle_example.wig
Converter settings to use
Default
Default settings should usually be used.
Run Tool
Help
Syntax
History
search datasets
Unnamed history
144 kB
2
3
2 : Wig/BedGraph-to-bigWig on data 1
44.4 KB
format bigwig, database Arabidopsis_thaliana_TAIR10
Binary UCSC BigWig file
1 : JBrowse_wiggle_example.wig

This tool converts bedgraph or wiggle data into bigWig type.

- **Wiggle format:** The .wig format is line-oriented. Wiggle data is preceded by a UCSC track definition line. Following the track definition line is the track data, which can be entered in three different formats described