Data Visualization in R Studio

Adapted from Jeff Berry's "R Bootcamp"

December 16, 2022

Parag Bhatt PhD
Data Science Core/ EROL

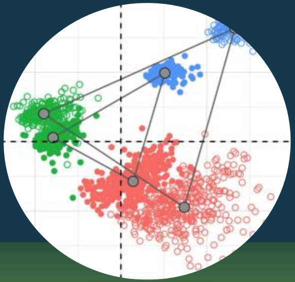
DONALD DANFORTH PLANT SCIENCE CENTER

Data Science

CLOUD COMPUTING

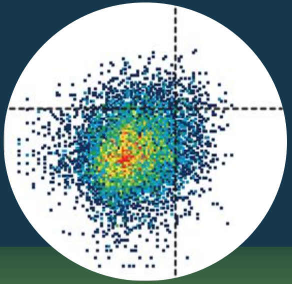
- High-performance computing and data storage
- Automated data management
- On-demand, flexible computational workflows

STATISTICAL ANALYSIS



- Experimental design and power analysis
- Parametric and non-parametric hypothesis testing
- Bayesian inference
- Dimensionality reduction
- Data visualization

GENOMICS & TRANSCRIPTOMICS

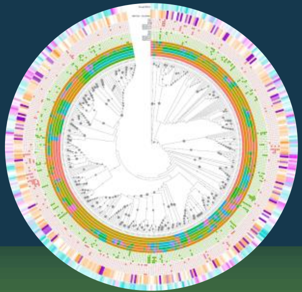


- Genome assembly and annotation
- Comparative genomics
- Differential expression analysis
- Gene network analysis

IMAGE ANALYSIS

- Image analysis workflows
- Machine learning
- Plant growth analysis

MICROBIOME



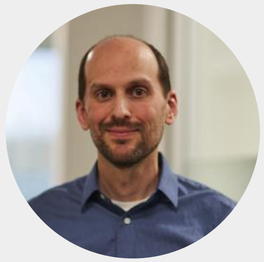
- Taxonomic classification
- Population diversity
- Differential analysis

SOFTWARE DEVELOPMENT

- Image analysis
- Computational workflow management
- Data management
- Interactive web applications

Contact: Noah Fahlgren, Director
Email: nfahlgren@danforthcenter.org

Data Science Team



NOAH FAHLGREN, *PhD, Director and Principal Investigator*

MAO LI, *PhD, Principal Investigator*

PARAG BHATT, *PhD, Data Science Trainer*

JORGE GUTIÉRREZ, *PhD, Postdoctoral Associate*

SETH POLYDORE, *PhD, Postdoctoral Associate*

JOSH ROTHHAUPT, *MS, Data Scientist*

HALEY SCHUHL, *MS, Data Scientist*

JOSH SUMNER, *MS, Data Scientist*

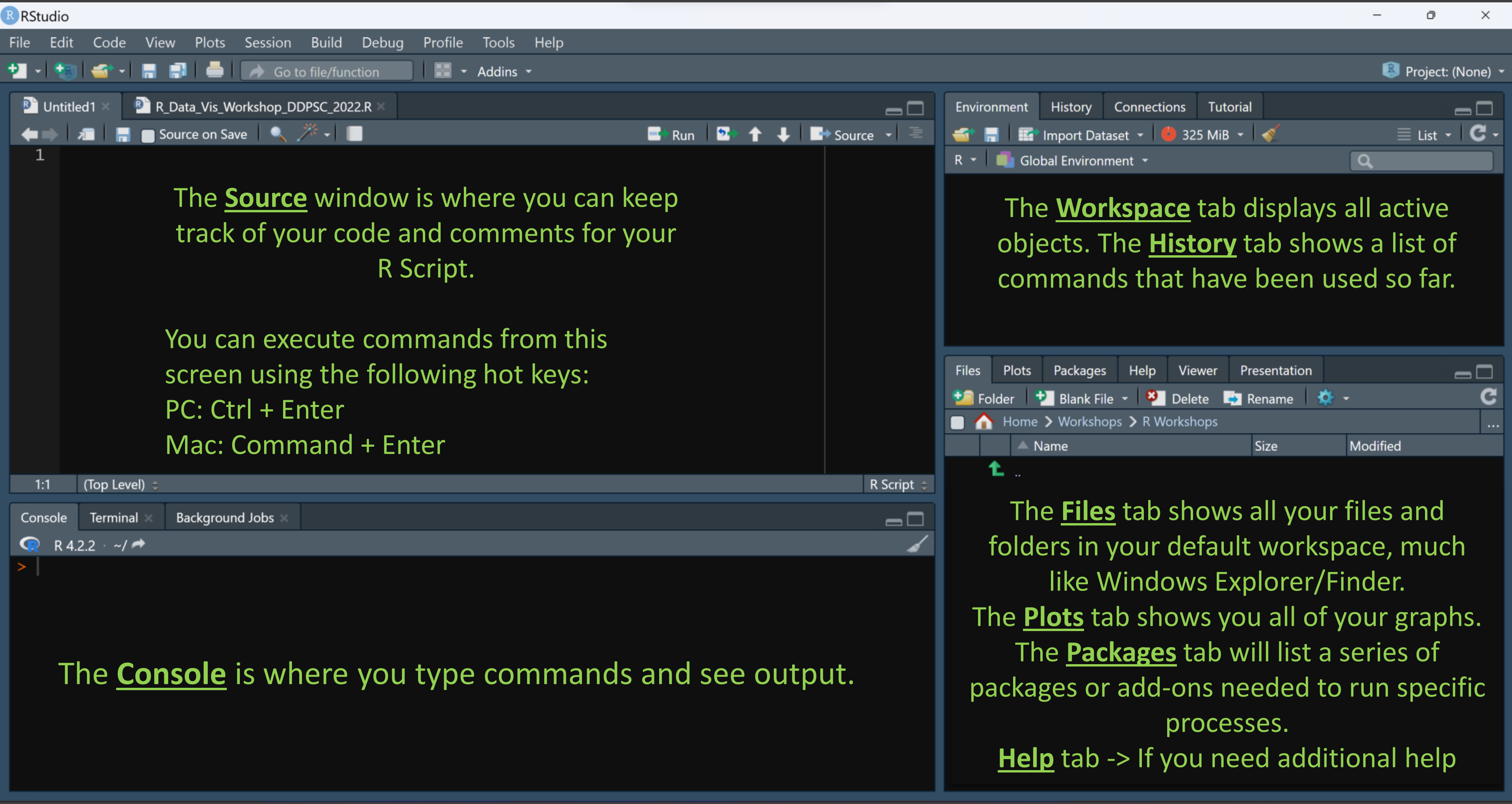
Learning Objectives:

- Understanding and navigating RStudio’s user interface
- Generating vectors, matrices, and data.frames
- Writing/reading/saving Excel files into RStudio
- Examining datasets and subsetting data
- Plotting single and multi-variate data using Tidyverse’s ggplot2 package

Why Do We Use R?

- Free, open-source, cross-platform
  - Large community following with free resources
  - Open-sourced packages available @ https://cran.r-project.org/
  - #help-datascience
- Scripts and packages are easily reproducible
- Flexibility in data handling
- Designed with data analysis in mind
  - Able to import and create datasets
- The user retains ultimate control over their data!

R Studio Layout


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Untitled1
R_Data_Vis_Workshop_DDPSC_2022.R
Source on Save
Run
Source
1:1
(Top Level)
R Script
Console Terminal Background Jobs
R 4.2.2 · ~/
Environment History Connections Tutorial
Import Dataset
325 MiB
List
R
Global Environment
Files Plots Packages Help Viewer Presentation
Folder Blank File Delete Rename
Home > Workshops > R Workshops
Name Size Modified

The **Source** window is where you can keep track of your code and comments for your R Script.

You can execute commands from this screen using the following hot keys:
PC: Ctrl + Enter
Mac: Command + Enter

The **Console** is where you type commands and see output.

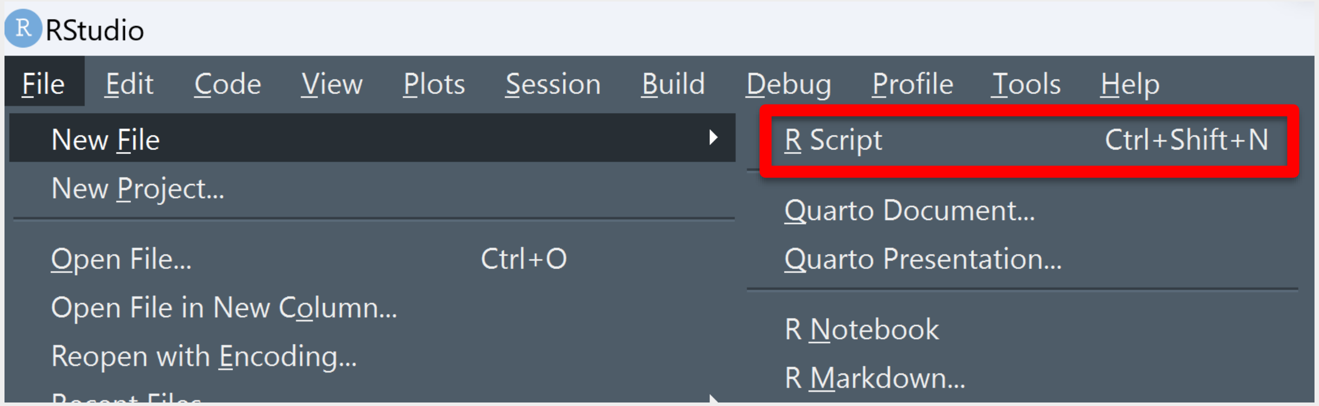
The **Workspace** tab displays all active objects. The **History** tab shows a list of commands that have been used so far.

The **Files** tab shows all your files and folders in your default workspace, much like Windows Explorer/Finder.
The **Plots** tab shows you all of your graphs.
The **Packages** tab will list a series of packages or add-ons needed to run specific processes.
Help tab -> If you need additional help

R Studio Layout

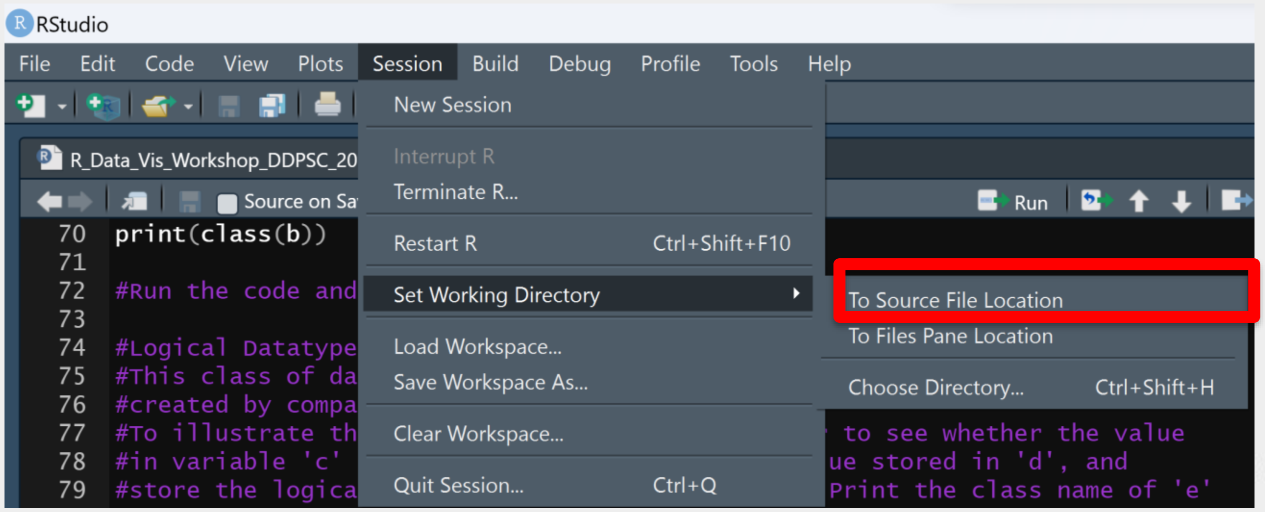
- **<u>Source window</u>** – keeps track of user code & comments
  - Create a new R script (see image below):
    - File > New > R Script
    - Click on icon with "+" sign > Select "R Script"
    - Ctrl + Shift + N

Always save your work!



R Studio Layout

- **Working Directory** – This is the folder that R looks for information (i.e., datasets, images, scripts, etc.).
  - If the data isn't in the working directory, then you will need the full file path to call the data
  - Changing your directory (see image below):
    - Ctrl + Shift +H
    - `setwd()`
    - `getwd() #Shows current working directory`



```
print(class(b))

#Run the code and

#Logical Datatype
#This class of da
#created by compa
#To illustrate th
#in variable 'c'
#store the logica
```

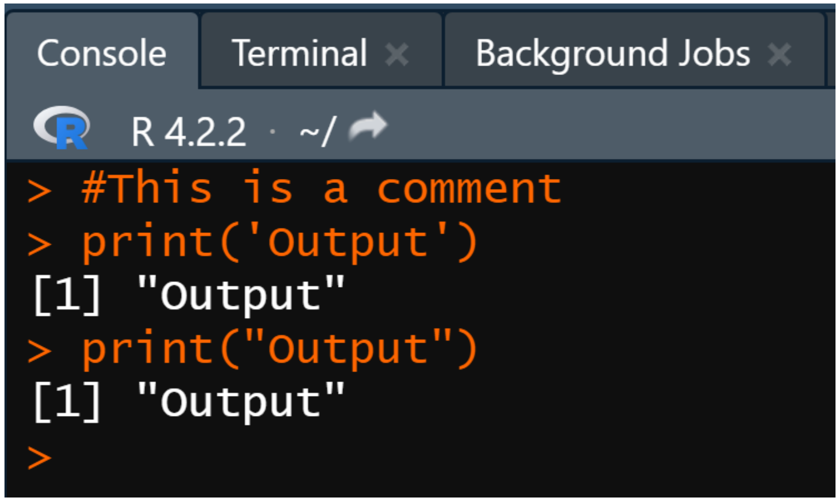
```
 to see whether the value
ue stored in 'd', and
Print the class name of 'e'
```

CENTER

Comments & Print Statements

- Use the *octothorpe* (**#**) symbol to place comments in your code – **a.k.a. *hashtag, pound sign, number sign, etc.***
- R will ignore anything that has **#** before it
- **`print()`** allows the user to display output in the console
  - If you want to display a string of text use single (‘’) or double (“”) quotation marks
- Give it a try(remember your hot keys for executing commands):
  - `#This is a comment`
  - `print(‘Output’)`
  - `print(“Output”)`

Does yours look like this?



```
> #This is a comment
> print('Output')
[1] "Output"
> print("Output")
[1] "Output"
>
```

Assigning Values to Objects

- Data can be stored in objects (variables)
- User-assigned identity to recall later for tasks
- Different approaches to storing data within objects:
  - `x <- 3`
  - `3->x`
  - `x=3`
- To recall the stored values, use:
  - `print(x)`
  - `x`

Handling Objects

- Using the previously stored “x”-value, perform the calculation:
  - `x+2  #Did you get 5?`
- Can also combine values to create vectors using the letter “c”, type the letter to recall the values
  - `s<-c(1,2,3) #Can store numbers`
  - `t<-c(“Fiji”, “Gala”, “Honeycrisp”) #Can store names`
  - `u<-c(u,4) #Can store vectors and values`
  - `v<-c(1,2)`

    `c(u,v)`

Data Classes

- R allocates memory based on the class of data stored in the object
- Exploring data classes:
  - `class(TRUE) #Logical class`
  - `class(4) #Numeric class`
  - `class(“Hello world!”) #Character class`
  - `class(c(1,2,3)) #Numeric class`
  - `class(matrix(c(1,2,3,4),nrow-2)) #Matrix and Array classes`
  - `class(data.frame(matrix(c(1,2,3,4),nrow=2))) #data.frame class`

Data Structures

- Most common data structures: Data Frames, Lists, and Vectors
  - Matrices are the most common data class used
- Two structures that allow you to mix different classes of objects into one:
  - data.frames
  - lists

Parsing Data in Vectors and Matrices

- Can specifically recall any element in a vector or matrix
  - R uses 1-based indexing

	H	E	L	L	O
0-based	0	1	2	3	4
1-based	1	2	3	4	5

- Vectors, use square brackets "[ ]":
  - `v<-c(1,2,3,4,5,6,7,8,9)`
  - `v[3] #Recalls element in the 3rd position`

Parsing Data in Vectors and Matrices

- Matrices, use `[row,col]` arrangement
  - `x<-matrix(seq(from=1, to=50, by=1),nrow=5)`
  - `x[1,3] #Extracts element stored in row 1, column 3 - 11`
- Collecting all elements of a row/column or a set of elements:
  - `x[1,] #Retrieves all columns of the 1st row`
  - `x[,3] #Retrieves all rows of the 3rd column`
  - `x[c(1,2),c(5,2)] #grabs elements (1,5),(1,2),(2,5),(2,2)`
  - `v[4:7] #grabs all elements whose index are between and including 4 and 7`

DONALD DANFORTH

PLANT SCIENCE CENTER

DANFORTHCENTER.ORG

Exploring Data Frames

- Load the dataset *`mtcars`*
  - Pre-loaded with R
- Grab the elements in the first column and return them as a vector:
  - Similar notation as parsing matrices
  - `mtcars[,1]`
- Returning column names of the dataset:
  - `colnames(mtcars)`
- Returning row names of the dataset:
  - `rownames(mtcars)`

Exploring Data Frames

- Extract the number of cylinders in the Honda Civic
  - `mtcars[“Honda Civic”, “cyl”]`
- Can also extract column information using “$”
  - `mtcars$hp`

Extracting Elements From Datasets

- Check to see if a car has six cylinder:
  - `mtcars$cyl==6`
- Check to see if a car does't have eight cylinders:
  - `mtcars$cyl!=8`
- Extract information for cars with 6 cylinders while retaining all the characteristics of the car into an object named `cyl6`
  - `cyl6<-mtcars[mtcars$cyl==6,]`
  - `cyl6`

Subsetting Our Data Further

- Extract all of the cars that have higher mpg than Volvo 12E and also have hp>80
  - `sub<-mtcars[mtcars$mpg>mtcars[“Volvo 142E”,”mpg”] & mtcars$hp>80,]`
  - `sub`
- Extract all of the cars that have either six cylinders or greater than 100 hp
  - `sub2 <- mtcars[mtcars$cyl>6 | mtcars$hp>100,]`
  - `sub2`

Other helpful functions call information

- What we’ve covered so far
  - `colnames(mtcars) #Calls column names of dataset`
  - `row.names(mtcars) #Calls row names of dataset`
- Other helpful functions
  - `str(mtcars) #Displays the internal structure of an object`
  - `dim(mtcars) #Displays the dimensions of the dataset`
  - `nrow(mtcars) #Calls the number of rows`
  - `ncol(mtcars) #Calls the number of columns`

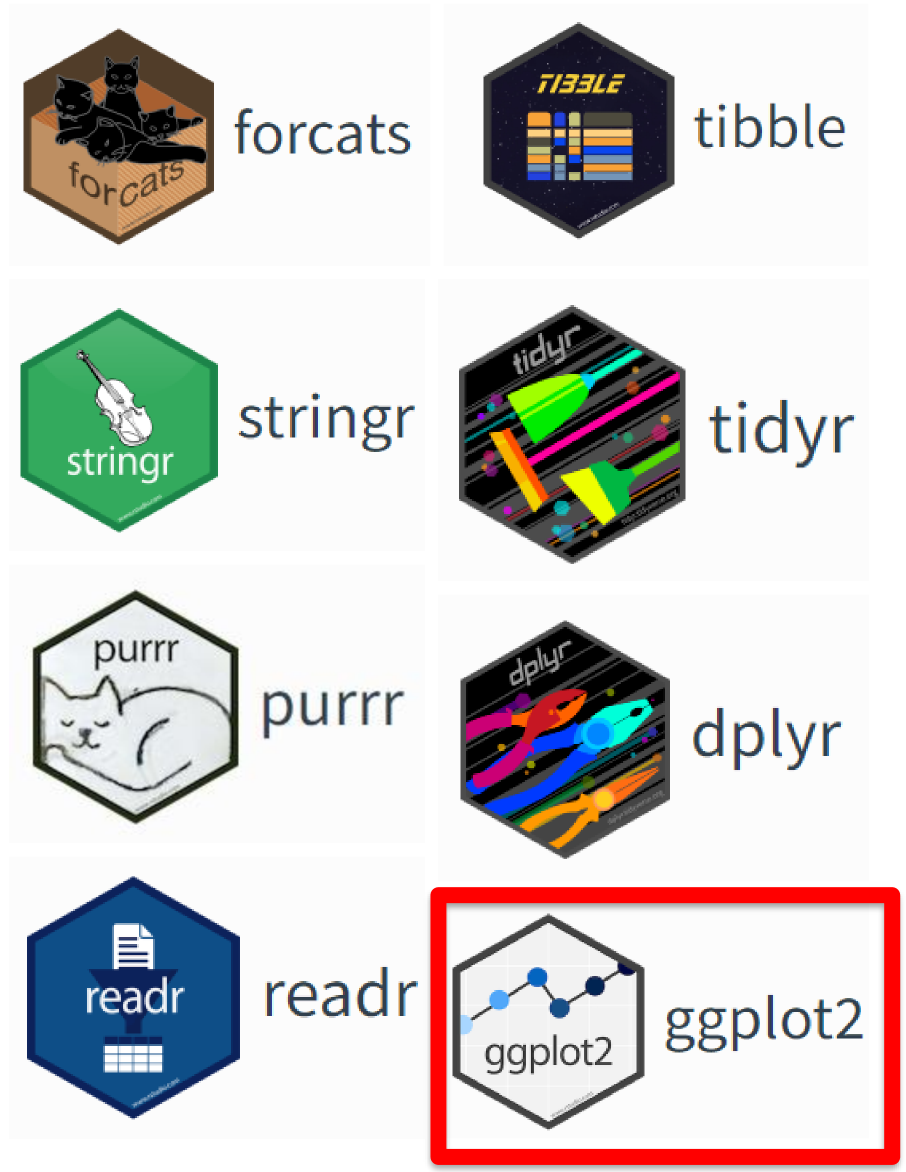
DONALD DANFORTH

PLANT SCIENCE CENTER

DANFORTHCENTER.ORG

Tidyverse

Integrated collection of R packages designed for data science



Importing Excel Files into R using readxl

- The `readxl` package simplifies importing data from Excel/Google Sheets and into the R environment.
  - supports .xls and .xlsx formats
- Installed as a part of Tidyverse, but loaded separately
  - `install.package(“tidyverse”)`
  - `library(tidyverse)`
  - `library(readxl)`

Importing Excel Files into R using readxl()

- To import a .xls/.xlsx file use the command:
  - `sv_shapes <- read_excel("sorghum_phenotyper_formatted.xlsx")`
- If you know the format of your spreadsheet, use `read_xls()` or `read_xlsx()`
- `excel_sheets()` allows you to read all of the sheets that are located within an Excel spreadsheet

Importing Excel Files into R using read.csv

- To import a .csv file use the command:
  - `sv_shapes <- read.csv(“sorghum_phenotyper_formatted.csv”,header=T,stringsAsFactors = F)`
- To write .csv files use the command:
  - Use the command `?write.csv()` to learn more about how to label your attribute descriptions.

Investigating Your Dataset

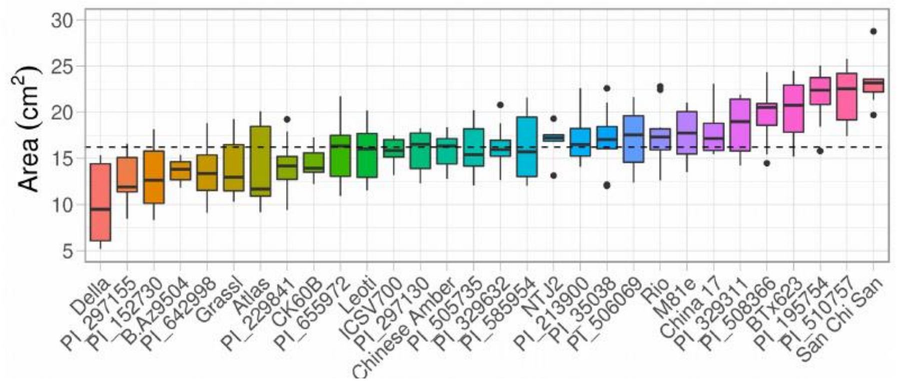
- You can investigate the dataset with the following functions:
  - `head()` – returns the first 6 lines in the dataset
  - `tail()` – returns the last 6 lines in the dataset
  - `summary()` – displays summary statistics for each column
  - `sum()` – calculate the sum of a vector's elements
  - `mean()` – calculates the mean of elements

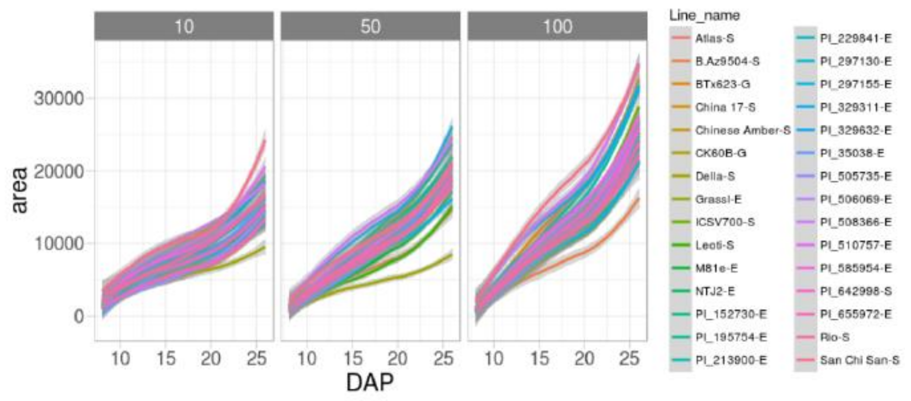
Interacting with Sorghum Dataset

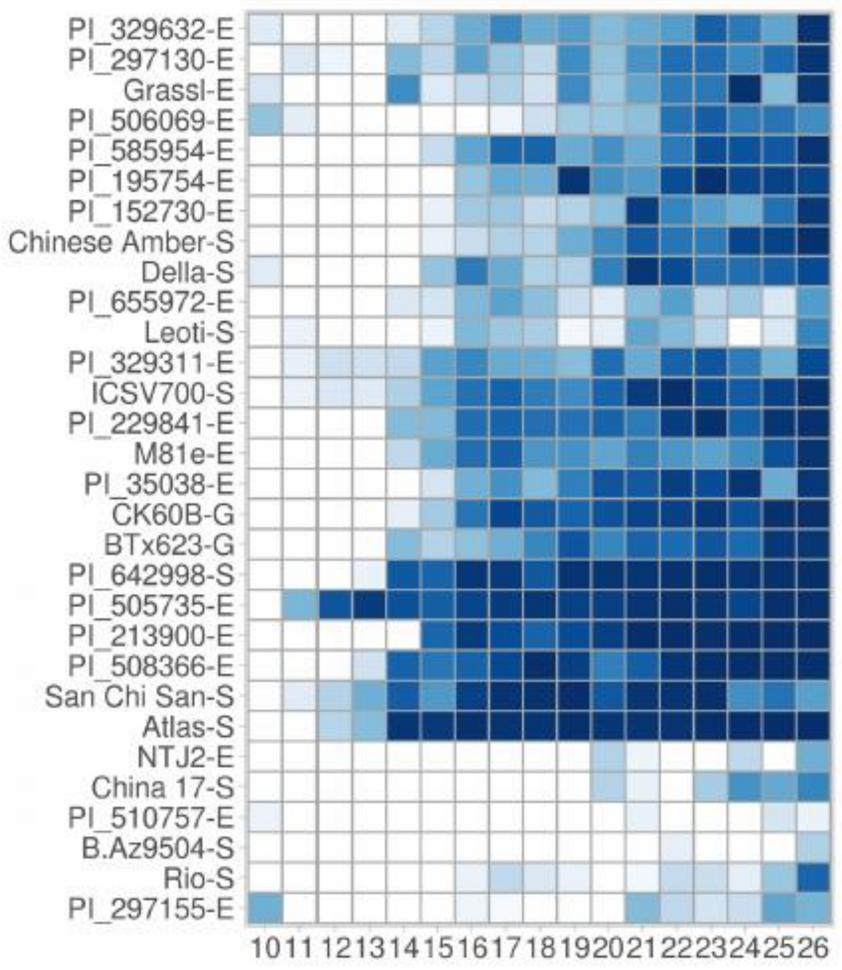
- Data was gathered on the phenotyper
- Analyzed using the Danforth's PlantCV software for plant phenotyping
- Different lines of sorghum were exposed to varying nitrogen treatments
- Questions for you:
  1. How would you determine the number of nitrogen treatments used in this experiment?
  2. What is the average area for the genotype "Della-S" on DAP 24?

Break Time – Meet back in 15 minutes

DONALD DANFORTH
PLANT SCIENCE CENTER


Area (cm^2)
30
25
20
15
10
5
Della
PI_297155
PI_152730
B.Az9504
PI_642998
Grassl
Atlas
PI_229841
CK60B
PI_655972
Leoti
ICSV700
PI_297130
Chinese Amber
PI_505735
PI_329632
PI_585954
NTJ2
PI_213900
PI_35038
PI_506069
Rio
M81e
China 17
PI_329311
PI_508366
BTx623
PI_195754
PI_510757
San Chi San


10
50
100
area
30000
20000
10000
0
10
15
20
25
DAP
Line_name
Atlas-S
B.Az9504-S
BTx623-G
China 17-S
Chinese Amber-S
CK60B-G
Della-S
Grassl-E
ICSV700-S
Leoti-S
M81e-E
NTJ2-E
PI_152730-E
PI_195754-E
PI_213900-E
PI_229841-E
PI_297130-E
PI_297155-E
PI_329311-E
PI_329632-E
PI_35038-E
PI_505735-E
PI_506069-E
PI_508366-E
PI_510757-E
PI_585954-E
PI_642998-S
PI_655972-E
Rio-S
San Chi San-S


PI_329632-E
PI_297130-E
Grassl-E
PI_506069-E
PI_585954-E
PI_195754-E
PI_152730-E
Chinese Amber-S
Della-S
PI_655972-E
Leoti-S
PI_329311-E
ICSV700-S
PI_229841-E
M81e-E
PI_35038-E
CK60B-G
BTx623-G
PI_642998-S
PI_505735-E
PI_213900-E
PI_508366-E
San Chi San-S
Atlas-S
NTJ2-E
China 17-S
PI_510757-E
B.Az9504-S
Rio-S
PI_297155-E
10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26

How would you make graphs on a paper?

1. Gather your graph paper
2. Draw and label axes
3. Insert data and adjust axes to accurately represent the data
4. Label groups
5. Add additional features (i.e., line of best-fit, etc.)

★ ggplot does the same thing when designing graphs

Let’s get started…

- Load ggplot2 package into your R environment
  - `library(ggplot2)`
- Set your working directory to a new folder so we keep our data organized

Getting started with ggplot2

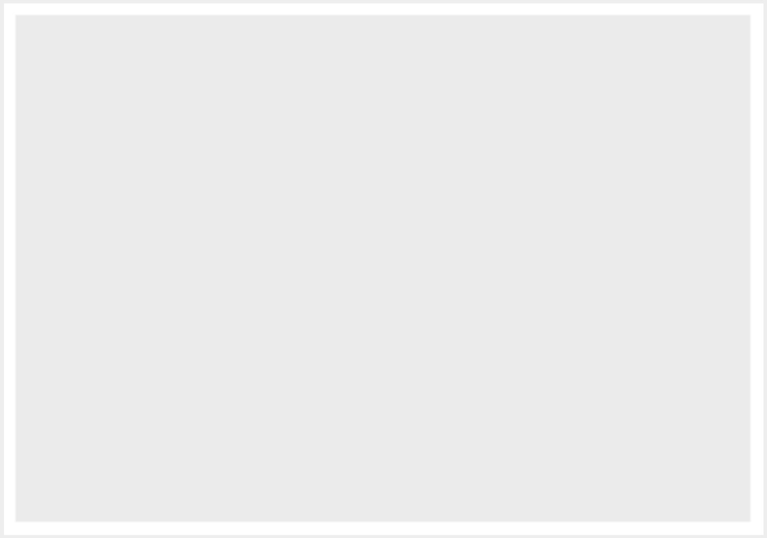
- Make sure you have the sorghum phenotyper data read into your environment.
  - `sv_shapes <- read_xlsx ("sorghum_phenotyper_formatted.xlsx")`
- Gather data from the last day of this experiment and store it in an object named `last_day`
  - `last_day <- sv_shapes[sv_shapes$DAP==max(sv_shapes$DAP),]`
- Get in the habit of investigating your subset data to make sure you have the information you need

Plotting Area vs. Perimeter on Last Day of Sorghum Phenotyper Experiment

- Investigate the data points of our data subset
  - `plot(data=last_day, area~perimeter)`
- Making figures with ggplot2
  - Make a blank figure stored into an object named "p"
  - p <- ggplot()
- Click on the ***plot*** tab to view this "figure"
  - Do you see a blank gray rectangle?

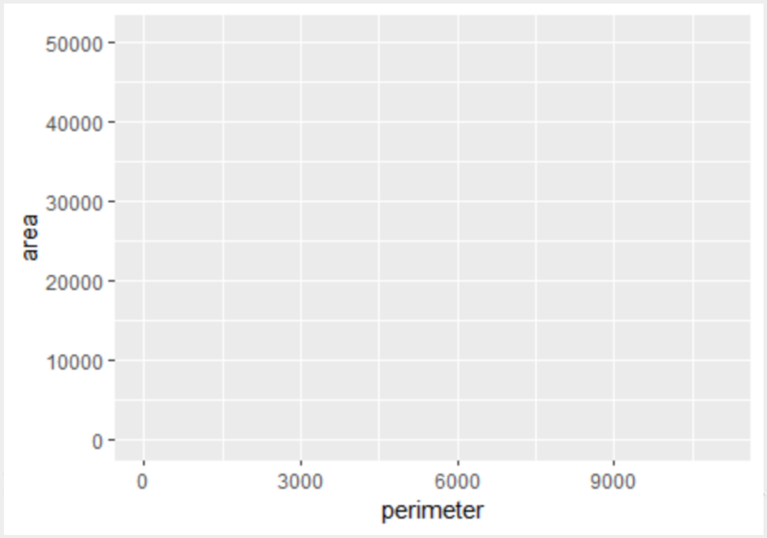
Plotting Area vs. Perimeter on Last Day of Sorghum Phenotyper Experiment

- Define the data that you are going to use:
  - `p <- ggplot(data=last_day)`



- Now, we establish the information on the x- and y-axes
  - `p <- ggplot(data=last_day, aes(perimeter, area))`

    x y



Plotting Area vs. Perimeter on Last Day of Sorghum Phenotyper Experiment

- Time to start adding data to the figure
  - At this stage, our object “p” is a called a **mapping**
  - Our numerical data will be transferred to the map using **layers**
    - geometries
    - scales
    - themes
- Next step is to start adding points to the figure, which is a <u>geometry</u> attribute

Plotting Area vs. Perimeter on Last Day of Sorghum Phenotyper Experiment

- "+" sign - Adds a layer to an existing map
  - `p <- ggplot(data=last_day, aes(perimeter, area))+geom_point()`
- Another method for adding layers

  -
    ```
    p <- ggplot(data=last_day, aes(perimeter, area)) +
    p <- p + geom_point()
    ```
- Second method facilitates on-the-fly updates

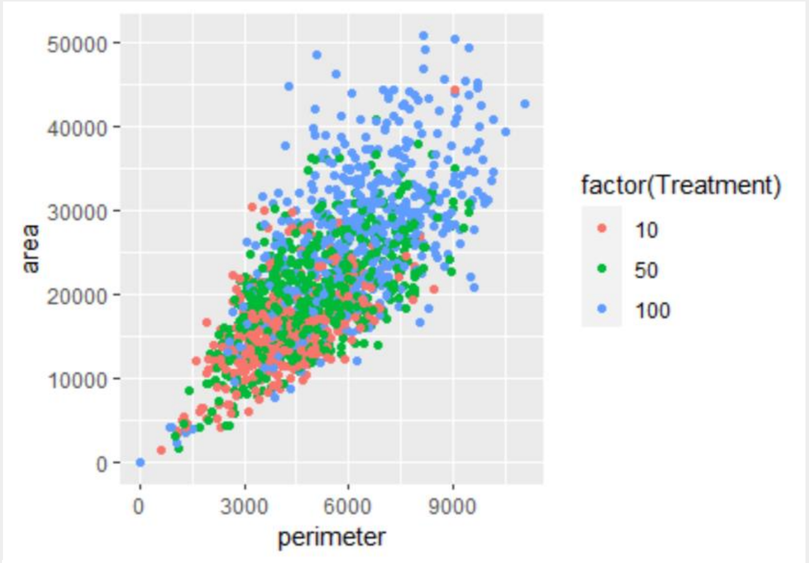
Plotting Area vs. Perimeter on Last Day of Sorghum Phenotyper Experiment

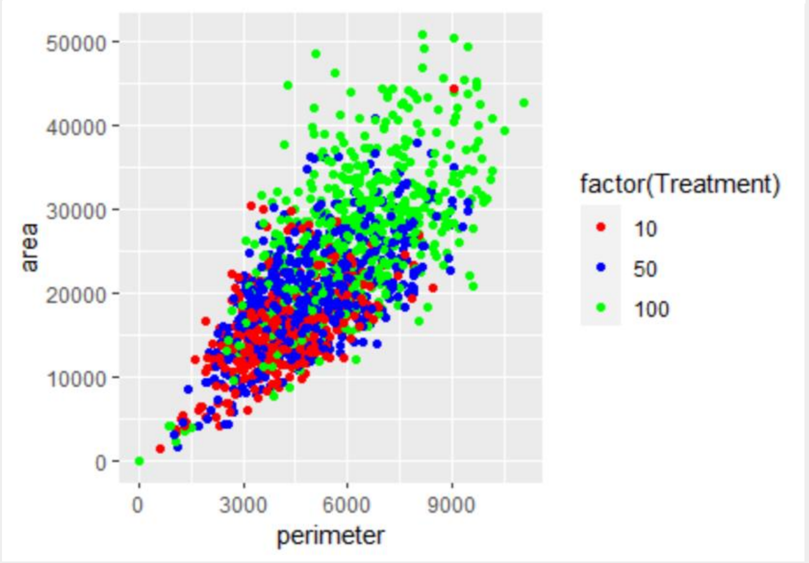
- Aesthetic attributes of figures
  - necessary for distinguishing data
  - makes graphs visually pleasing
- For our data, we will color code our nitrogen treatments

  -
    ```
    p <- ggplot(data=last_day,
    aes(perimeter,
    area))+geom_point(aes(color=factor(Tr
    eatment)))
    ```

Scale Layers

- The default colors are neat, but you can customize the data points to put your own flavor on the figure
- We can alter the color of our data by adding a scale layer to change the color aesthetic
  - p <- ggplot(data=last_day,aes(perimeter,area))+ geom_point(aes(color=factor(Treatment)))+ scale_color_manual(values = c("red","blue","green“))
- For more information on scales, visit this reference guide





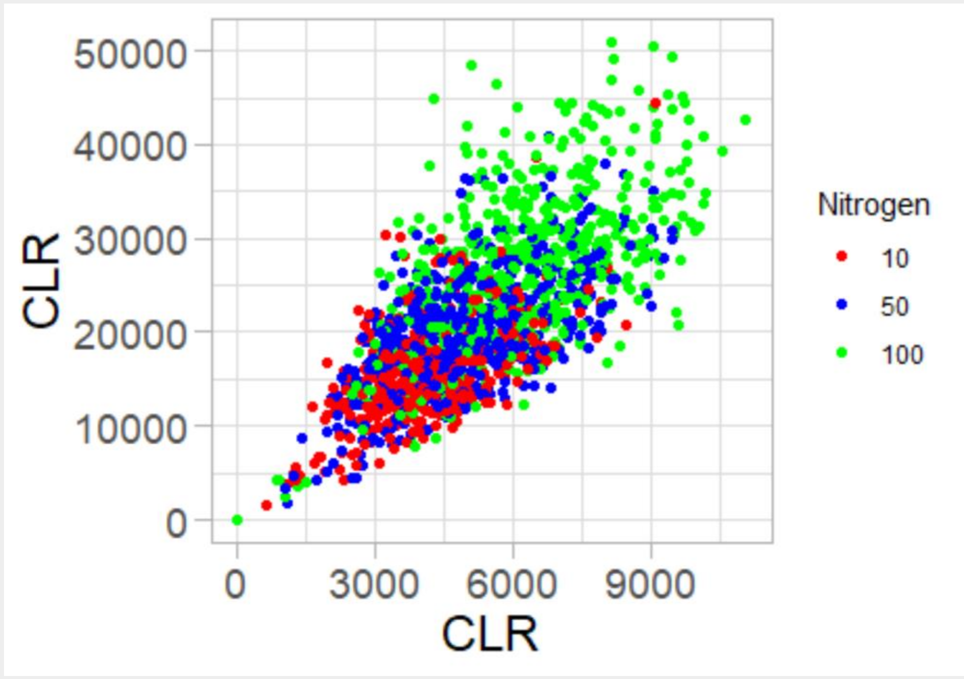
Adding Theme Layers to our Plot

- Now we will add theme layers to our figure to further distinguish it by alter the appearance of the figure’s layout
- For more information on how to use theme’s to customize your figure layout, visit this reference document

Plotting Area vs. Perimeter on Last Day of Sorghum Phenotyper Experiment

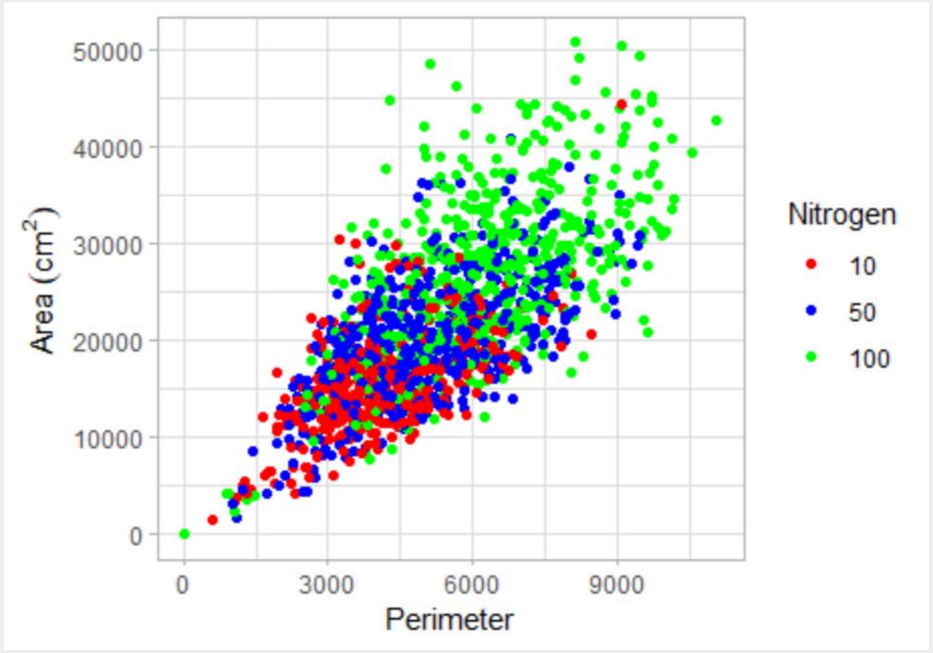
Append your code to include the following themes:

```
xlab("CLR")+
ylab("CLR")+
theme_light()+
theme(strip.background=element_rect(f
ill="gray50"),
strip.text.x=element_text(size=14,col
or="white"),
strip.text.y=element_text(size=14,col
or="white"))+
theme(axis.title= element_text(size =
18))+
theme(axis.text = element_text(size =
14))+
theme(axis.ticks.length=unit(0.2,"cm"
))+
guides(color = guide_legend(title =
"Nitrogen"))
```



Plotting Area vs. Perimeter on Last Day of Sorghum Phenotyper Experiment

What changes would you make to alter the labels on the x- and y-axis to reflect the figure below?



Lastly, we need to save our plots. Otherwise, what's the point?

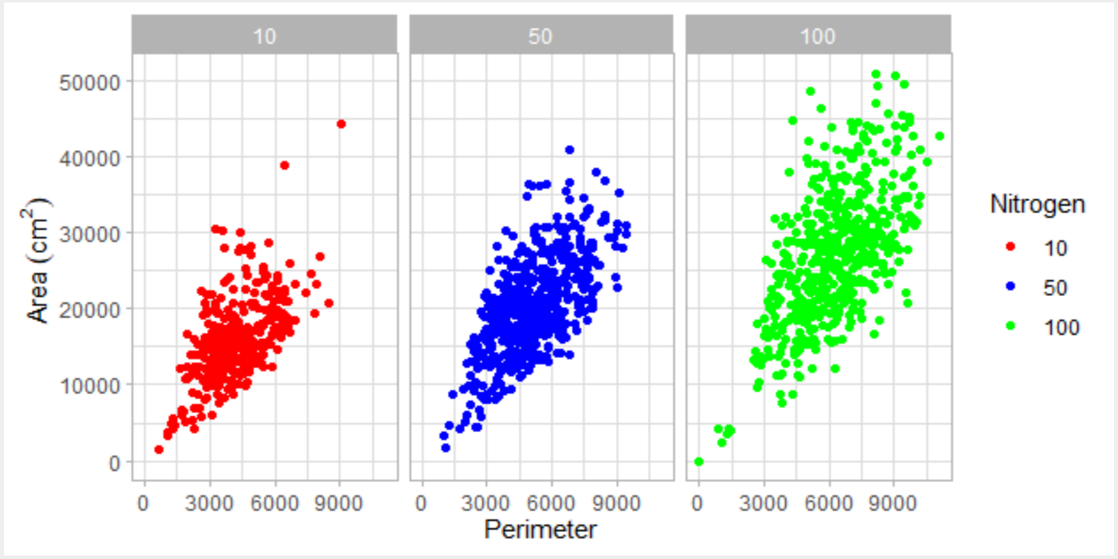
```
ggsave("sorghum_nitrogen_perimeter_vs_area.png", width=8.91, height=4.55, plot=p,dpi=300)
```

Faceting

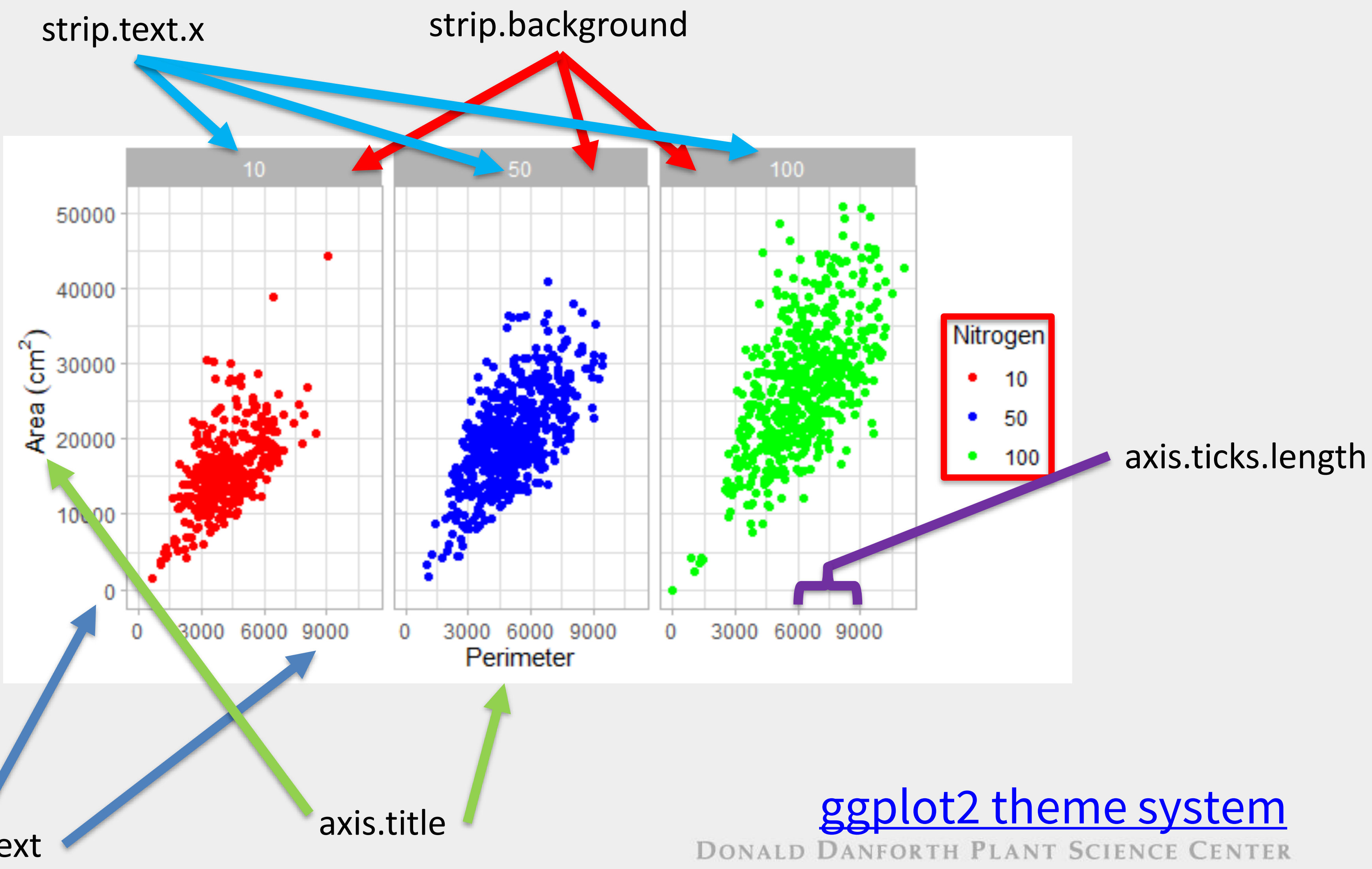
- Generates small multiples for displaying different subsets of data
  - Small multiples allow for quick comparison of the data
- Three types of faceting:
  1. `facet_null()` – a single plot, the default
  2. `facet_wrap()` – “wraps” a 1d ribbon of panels into 2d; useful if you have a single variable with many levels and want to arrange the plots in a more space efficient manner
  3. `facet_grid()` – produces a 2d grid of panels defined by variables which form the rows and columns
- We are going to use faceting to separate our scatter plot so we can more easily visualize differences between nitrogen treatments

Faceting Data Points

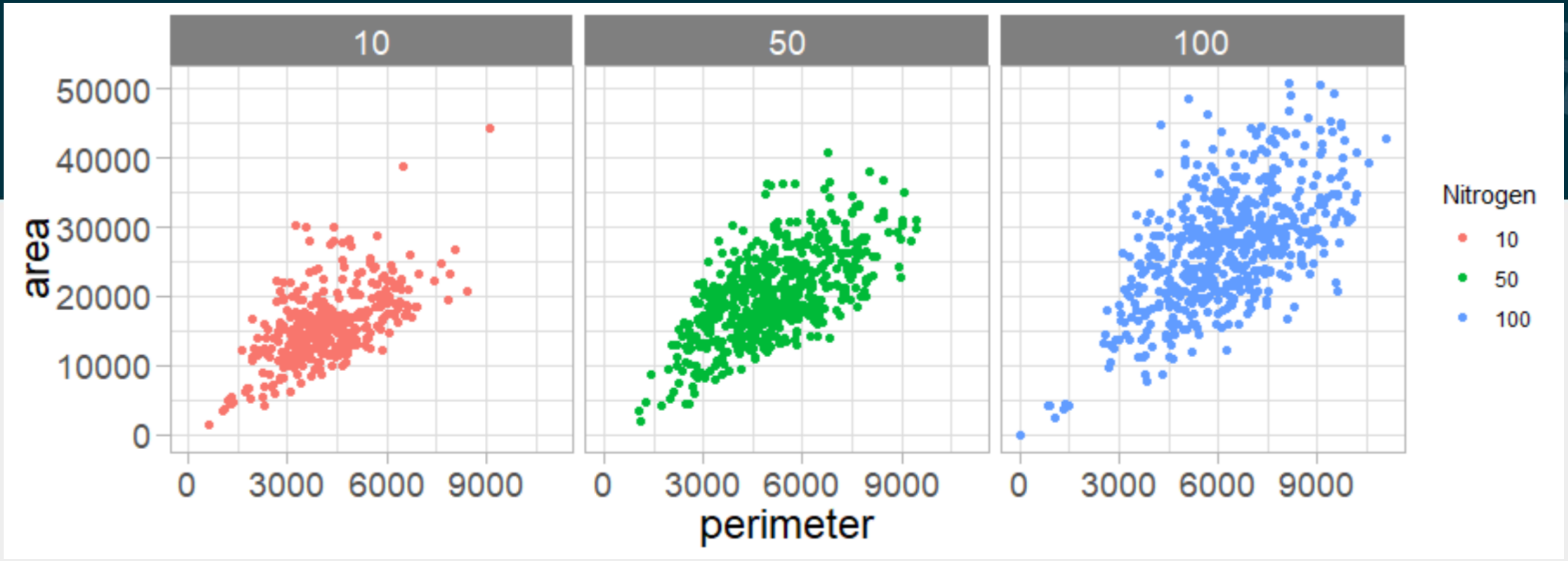
- Using `facet_grid()` separate our nitrogen treatments across 3 different small plots.
  -
```
p <- ggplot(data=last_day,aes(perimeter,area))+
  facet_grid(~Treatment)+
  geom_point(aes(color=factor(Treatment)))+
  scale_color_manual(values=c("red","blue","green"))+
  heme_light()+
  xlab("Perimeter")+
  ylab(~~Area~(cm^2))+
  guides(color=guide_legend(title = "Nitrogen"))
```



Themes - Altering Panel Strip Attributes

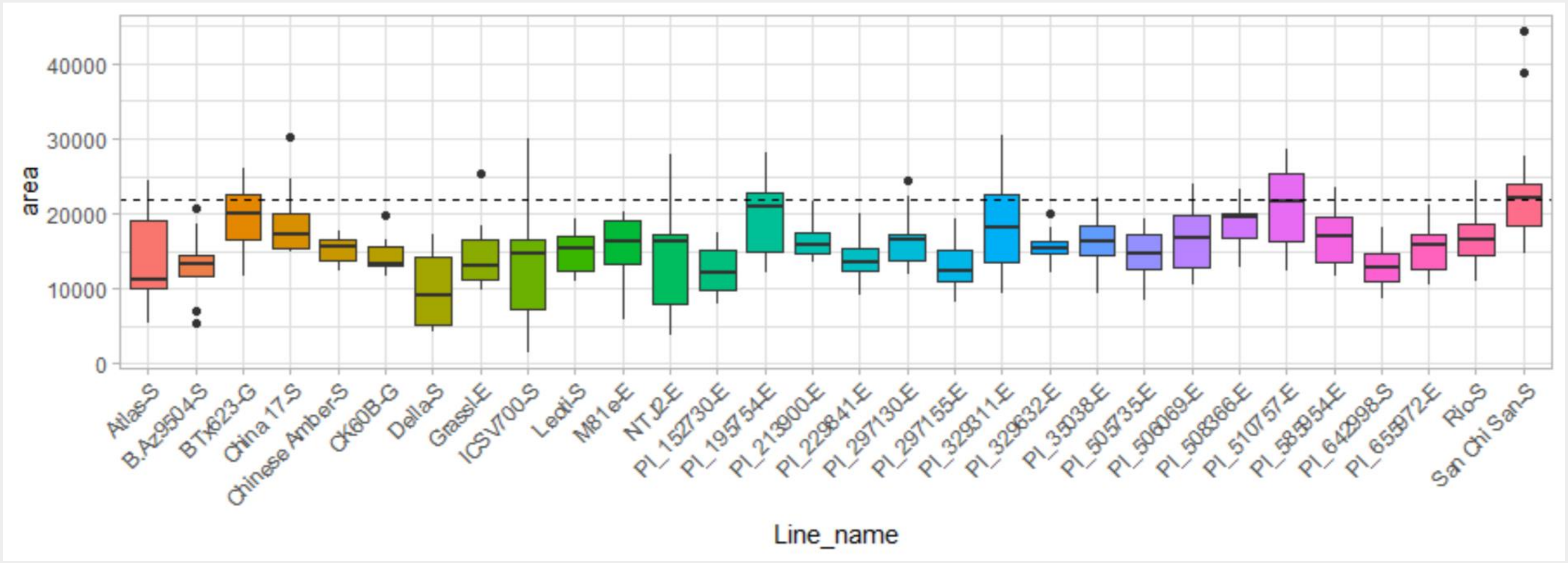


ggplot2 theme system



```
• p <- ggplot(data=last_day, aes(perimeter,area))+
       facet_grid(~Treatment)+
       geom_point(aes(color=factor(Treatment)))+
       theme_light()+
       theme(strip.backround=element_rect(fill="gray50"),
              strip.text.x=element_text(size=14,color="white"),
              strip.text.y=element_text(size=14,color="white"))+
       theme(axis.title=element_text(size=18))+
       theme(axis.text=element_text(size=14))+
       theme(axis.ticks.length=unit(0.2,"cm"))+
       guides(color=guide_legend(title="Nitrogen"))
```

ggplot2 Challenge



Use only the data subset for `treatment=10`

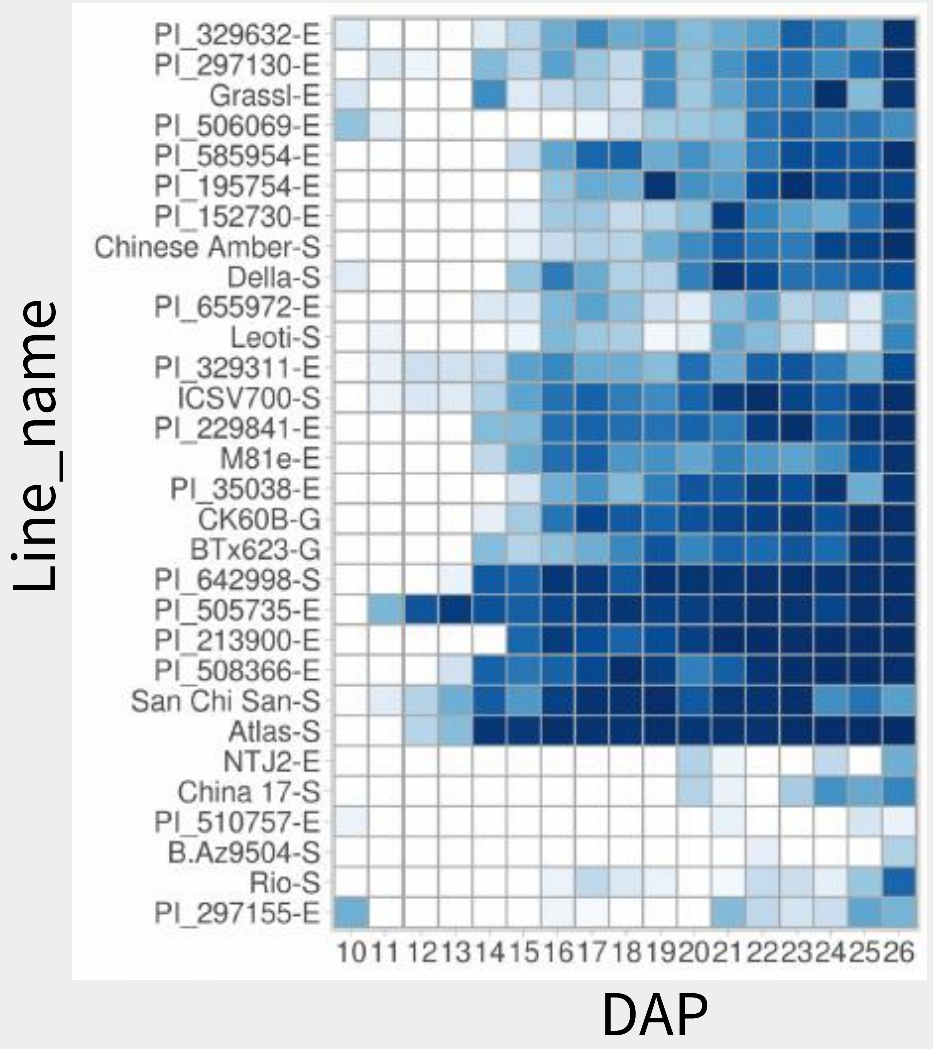
Use the following attributes: `geom_hline(), geom_boxplot(), theme(axis.text.x = element_text(angle = 45, hjust = 1)), ylab, and xlab`

Break Time – Meet back in 15 minutes

DONALD DANFORTH

PLANT SCIENCE CENTER

Plot Multiple Variables



Plotting Mean Sorghum Plant Area By Line_name and Day after Planting (DAP)

Objective: Expand the figure on the previous slide to illustrate how Nitrogen treatments affect mean plant area.

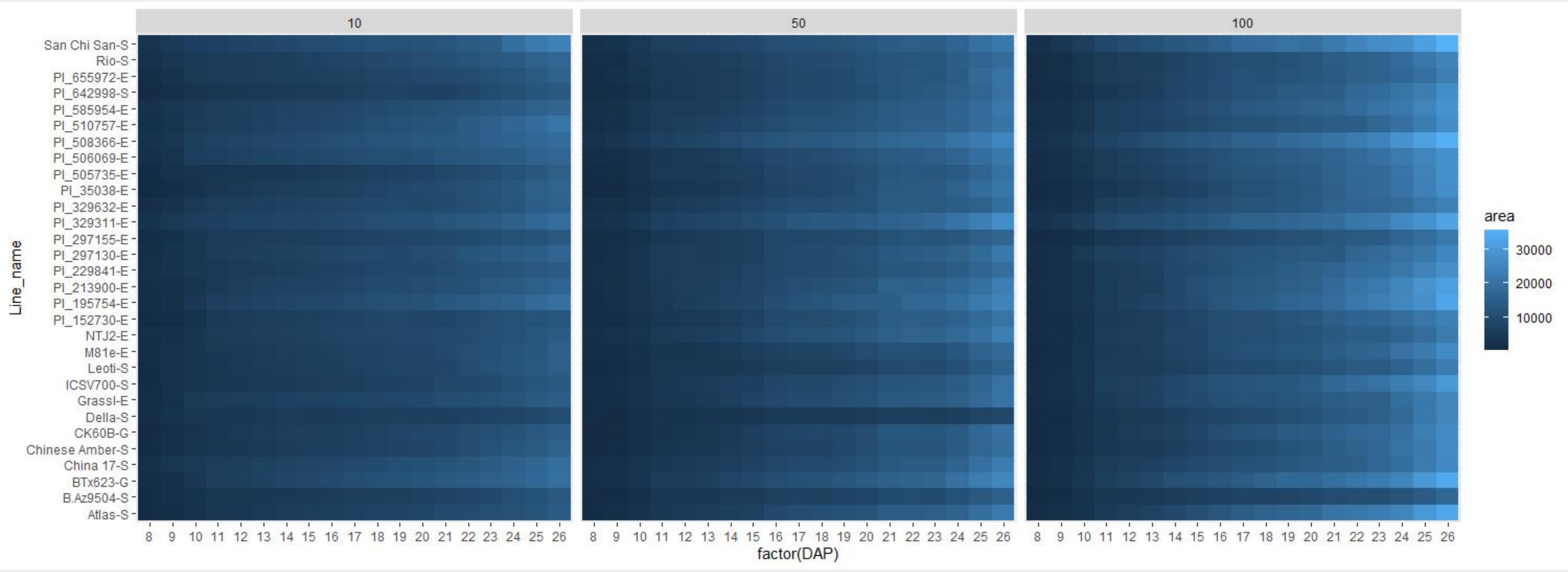
```
my_df <-
aggregate(data=sv_shapes,area~Line_
name:Treatment:DAP,FUN = "mean")
```

Plotting Mean Sorghum Plant Area By Line_name and Day after Planting (DAP)

Objective: Expand the figure on the previous slide to illustrate how Nitrogen treatments affect mean plant area.

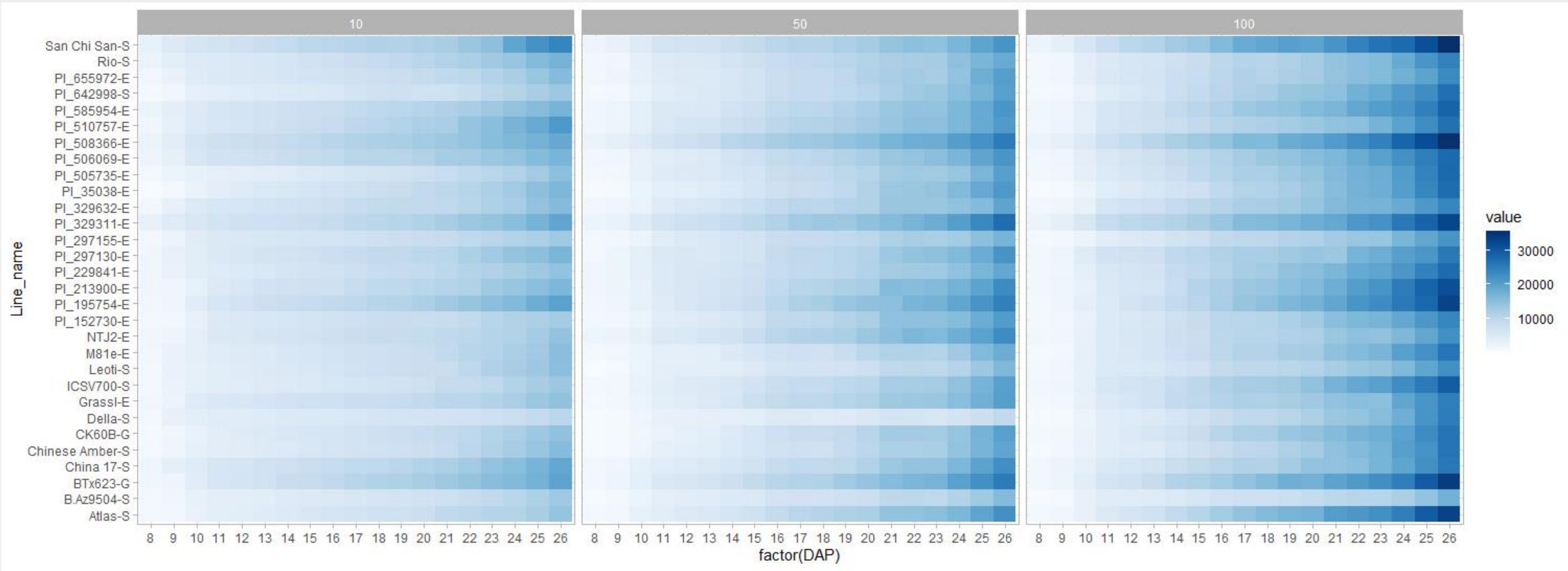
Goal: A figure that displays the mean plant area per genotype based on DAP. The figure should be split into three separate grids to account for each nitrogen treatment in the experiment.

Plotting Mean Sorghum Plant Area By Line_name and Day after Planting (DAP)



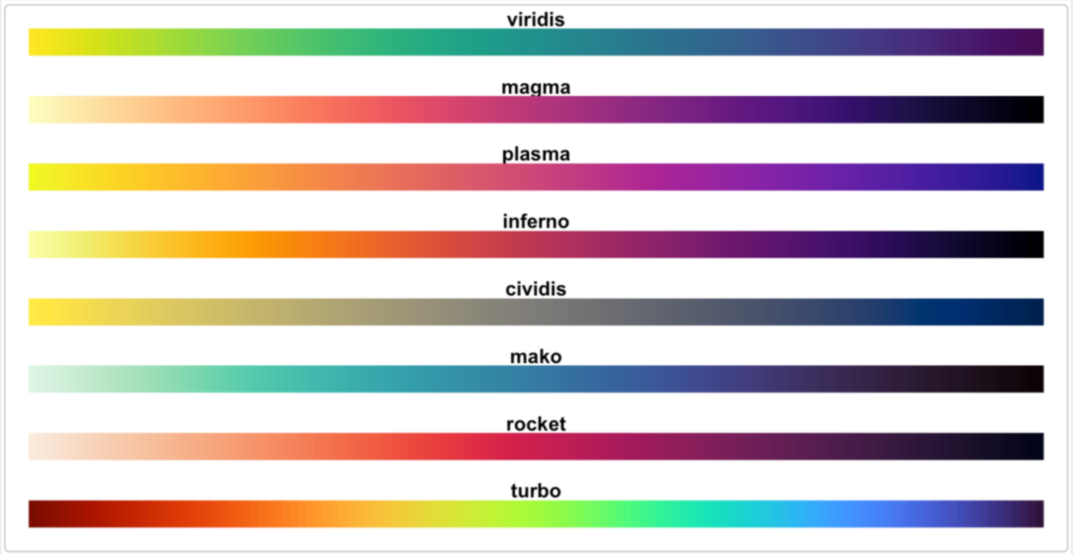
Changing Color Schemes

```
library(RColorBrewer)
q <- ggplot(data=my_df,aes(DAP,Line_name))+
           facet_grid(~Treatment)+
           geom_tile(aes(fill=area))+
   scale_fill_gradientn(colors=(colorRampPalette(brewer.pal(9,"Blue
s"))(10)),"value")+ theme_light()
```

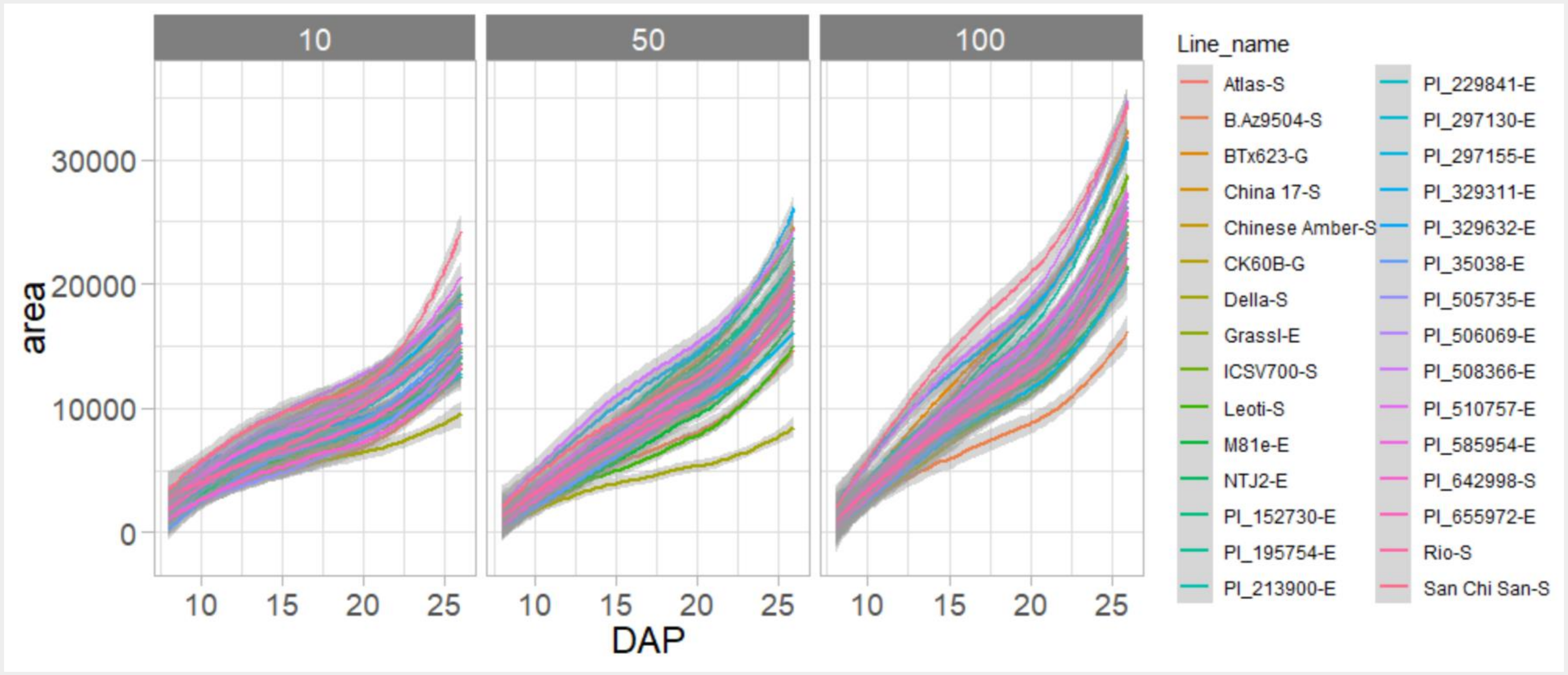


Other Color Palettes

- Viridis color
  - install.packages(“viridis”)
- ggplot2 colors
  - Comes preloaded in ggplot2 package
- Color Lisa
  - Repository of color palettes from famous artists
  - install.packages(“lisa”)
- Scientific journal color palettes
  - install.packages(“ggsci”)
- Wes Anderson color palettes
  - install.packages(“wesanderson”)
- R base color palettes
  - Comes preloaded with R
- R Charts color palettes



ggplot2 Challenge



Helpful Websites & Resources

- https://ggplot2.tidyverse.org/
- https://ggplot2-book.org/index.html
- https://www.sharpsightlabs.com/blog/ggplot2-tutorial/
- https://www.rdocumentation.org/packages/ggplot2/versions/3.3.5
- https://stackoverflow.com/
- https://cran.r-project.org/web/packages/ggplot2/ggplot2.pdf
- https://datacarpentry.org/R-ecology-lesson/04-visualization-ggplot2.html
- Slack → #help-datascience

Commitment to an Inclusive and Safe Work Environment

The Donald Danforth Plant Science Center is committed to creating and maintaining a diverse, equitable, inclusive, respectful and safe environment.

- We actively welcome diverse people, cultures, and perspectives.
- We strive to provide an environment in which everybody feels comfortable and excels.
- We do not tolerate discrimination or harassment of any kind.

If you experience or witness such behavior, please notify Human Resources, your supervisor or a member of the Danforth Center leadership team.

Our anonymous Ethics Hotline (1-800-455-0550) is also available for reporting concerns. Calls are received 24 hours a day, 7 days a week.

Fin.

DONALD DANFORTH
PLANT SCIENCE CENTER

https://survey.iad1.qualtrics.com/jfe/form/SV_6KlRnNOPHaVeBPE