

HW solutions

Md Rasheduzzaman

2025-09-09

dataframe, matrices, list, factor, vector, tidyverse, etc.

Table of contents

1	L3: Matrices and Lists	2
1.1	Task 1: Protein Quantification in Biological Samples	2
1.1.1	1. Matrices	2
1.1.2	2. Making transpose of ProteinMatrix	3
1.1.3	3. Identity Matrix	4
1.1.4	4.1. Total Protein per Sample	4
1.1.5	4.2. Total Protein per Protein Type	4
1.1.6	4.3. Heatmap of Protein Concentrations	4
1.1.7	Interpretation	5
1.2	Task 2: Gene-to-Protein Translation	6
1.2.1	Protein Output	7
1.2.2	2. Transpose	7
1.2.3	3. Identity matrix and multiplication	7
1.2.4	4. Sub-matrix:	8
1.2.5	Visualization tasks:	8
1.2.6	Interpretation	11
1.3	Task 3: Animal Breeding – Bull Ranking by Economic Traits	11
1.3.1	Define Data	11
1.3.2	Compute Total Economic Value	12
1.3.3	Multiply with Identity Matrix	12
1.3.4	Remove Milk Yield and Recalculate Total Value	13
1.3.5	Bar Plot: Total Economic Value	13
1.3.6	Heatmap of EBVs	14
1.3.7	Interpretation Questions	15
1.4	Task 4: Plant Breeding – Trait Contributions from Parental Lines	16
1.4.1	Define Data	16

1.4.2	Compute HybridTraits Vector	16
1.4.3	Multiply with Identity Matrix	17
1.4.4	Remove T3 (Maturation Time) and Recalculate	17
1.4.5	Heatmap of ParentTraits	18
1.4.6	Bar Plot of Hybrid Traits	19
1.4.7	Interpretation Questions	20
1.5	Task 5: Managing Matrices and Weight Vectors Using Lists in R	21
1.5.1	Create a Master List	21
1.5.2	List the Structure	22
1.5.3	Indexing Elements from Lists	23
1.5.4	Perform Weighted Calculations	24
1.5.5	Subset and Recalculate	25
1.5.6	Visualization Tasks	25
1.5.7	Interpretation Questions	28
1.6	Homework solutions: Factors, Subsetting, and Biological Insight	28
1.7	Homework Solution tidyverse	33

1 L3: Matrices and Lists

1.1 Task 1: Protein Quantification in Biological Samples

We measured the concentration (in $\mu\text{g}/\mu\text{L}$) of three proteins (P1, P2, P3) in four samples (S1–S4):

1.1.1 1. Matrices

```
# Making Protein Matrix
ProteinMatrix <- matrix(
  c(5, 3, 2,
    7, 6, 4),
  nrow = 2, byrow = TRUE
)
rownames(ProteinMatrix) = c("Sample1", "Sample2")
colnames(ProteinMatrix) = c("ProteinX", "ProteinY", "ProteinZ")
ProteinMatrix
```

	ProteinX	ProteinY	ProteinZ
Sample1	5	3	2
Sample2	7	6	4

Now goes the weight matrix

```
# Making weight matrix
WeightVector <- matrix(
  c(0.5, 1.0, 1.5),
  nrow=3, byrow = TRUE
)
rownames(WeightVector) = c("ProteinX", "ProteinY", "ProteinZ")
colnames(WeightVector) = c("Weight")
WeightVector
```

	Weight
ProteinX	0.5
ProteinY	1.0
ProteinZ	1.5

Now, multiply them.

```
# Multiplying Matrices
TotalConc = ProteinMatrix %*% WeightVector
colnames(TotalConc) <- "Total_Protein_Conc"
print(TotalConc)
```

	Total_Protein_Conc
Sample1	8.5
Sample2	15.5

1.1.2 2. Making transpose of ProteinMatrix

```
ProteinMatTranspose = t(ProteinMatrix)
ProteinMatTranspose
```

	Sample1	Sample2
ProteinX	5	7
ProteinY	3	6
ProteinZ	2	4

1.1.3 3. Identity Matrix

```
I <- diag(3)
Identitycheck = ProteinMatrix %*% I
colnames(Identitycheck) <- c("ProteinX", "ProteinY", "ProteinZ")
Identitycheck
```

	ProteinX	ProteinY	ProteinZ
Sample1	5	3	2
Sample2	7	6	4

1.1.4 4.1. Total Protein per Sample

```
rowSums(ProteinMatrix)
```

Sample1	Sample2
10	17

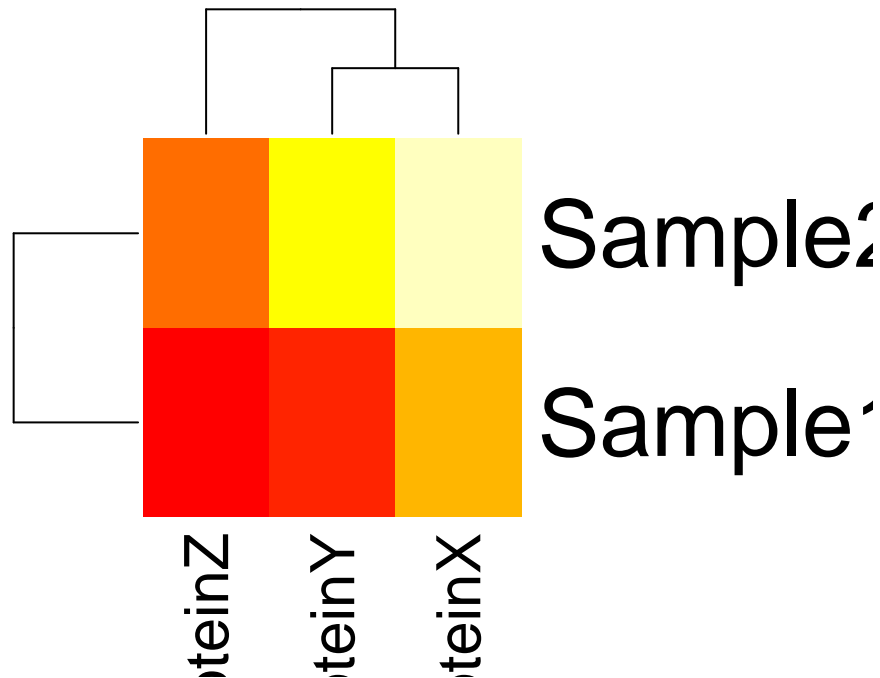
1.1.5 4.2. Total Protein per Protein Type

```
colSums(ProteinMatrix)
```

ProteinX	ProteinY	ProteinZ
12	9	6

1.1.6 4.3. Heatmap of Protein Concentrations

```
heatmap(ProteinMatrix, scale = "none", col = heat.colors(10))
```



1.1.7 Interpretation

- Multiplying the protein levels by the weight vector shows how much each protein contributes in a sample. The result shows total protein concentration per sample.
- The result shows that sample S2 has the highest protein burden.
- The identity matrix represents no protein interactions or measurement biases. It is a simple matrix calculation.
- New calculation:

```
# changing the weight of ProteinZ to 3.0
newweightvector = matrix(
  c(0.5, 1.0, 3.0),
  nrow=3, byrow = TRUE
)
rownames(WeightVector) = c("ProteinX", "ProteinY", "ProteinZ")
colnames(WeightVector) = c("Weight")
newTotalconc = ProteinMatrix %*% newweightvector
colnames(newTotalconc) <- "Total_Protein_Conc"
newTotalconc
```

```
      Total_Protein_Conc
Sample1              11.5
```

Sample2 21.5

Still, S2 has more protein burden.

Bonus:

- Heatmap reveals PX is most abundant across all samples.

1.2 Task 2: Gene-to-Protein Translation

```
# making Gene Expression matrix
GeneExpression <- matrix(
  c(10, 8, 5,
    15, 12, 10),
  nrow = 2, byrow = TRUE
)
rownames(GeneExpression) <- c("Sample1", "Sample2")
colnames(GeneExpression) <- c("GeneA", "GeneB", "GeneC")
GeneExpression
```

	GeneA	GeneB	GeneC
Sample1	10	8	5
Sample2	15	12	10

Translation efficiency:

```
# making Translation Matrix
TranslationMatrix <- matrix(
  c(1.5, 0 , 0,
    0, 1.2, 0,
    0, 0, 1.8),
  nrow = 3, byrow = TRUE
)

rownames(TranslationMatrix) <- c("GeneA", "GeneB", "GeneC")
colnames(TranslationMatrix) <- c("protA", "protB", "protC")
TranslationMatrix
```

	protA	protB	protC
GeneA	1.5	0.0	0.0
GeneB	0.0	1.2	0.0
GeneC	0.0	0.0	1.8

1.2.1 Protein Output

```
# computing Protein matrix
Protein_matrix <- GeneExpression %*% TranslationMatrix
colnames(Protein_matrix) <- c("total_protA", "total_protB", "total_protC")
print(Protein_matrix)
```

	total_protA	total_protB	total_protC
Sample1	15.0	9.6	9
Sample2	22.5	14.4	18

1.2.2 2. Transpose

```
# Transpose of GeneExpression matrix
GeneExpression_Transpose <- t(GeneExpression)
GeneExpression_Transpose
```

	Sample1	Sample2
GeneA	10	15
GeneB	8	12
GeneC	5	10

The new matrix represents a matrix where the rows and columns of GeneExpression matrix have been interchanged.

1.2.3 3. Identity matrix and multiplication

```
# Creating Identity matrix
I <- diag(3)
I
```

	[,1]	[,2]	[,3]
[1,]	1	0	0
[2,]	0	1	0
[3,]	0	0	1

Now, multiply:

```
Product_matrix = TranslationMatrix %*% I
Product_matrix
```

```
      [,1] [,2] [,3]
GeneA  1.5  0.0  0.0
GeneB  0.0  1.2  0.0
GeneC  0.0  0.0  1.8
```

The product is identical to TranslationMatrix

1.2.4 4. Sub-matrix:

```
# making submatrix A
A = matrix(
  c(10, 8,
    15, 12), nrow=2, byrow = TRUE
)
rownames(A) = c("sample1", "sample2")
colnames(A) = c("GeneA", "GeneB")

A
```

```
      GeneA GeneB
sample1    10     8
sample2    15    12
```

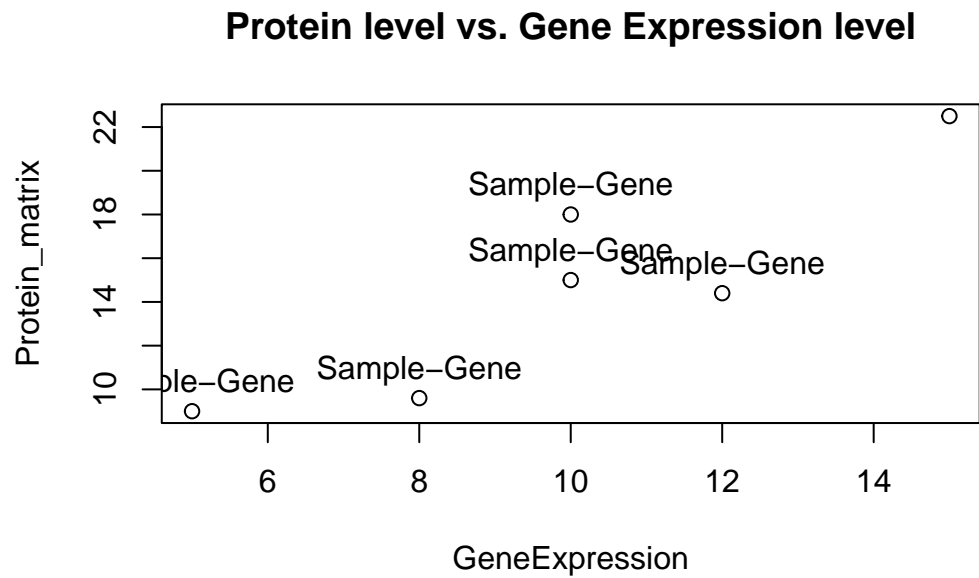
```
# finding inverse of A
#inv_A <- solve(A)
#inv_A
```

The inverse matrix could not be calculated since A is a singular matrix. So, $A * A^{-1}$ is also not possible.

1.2.5 Visualization tasks:

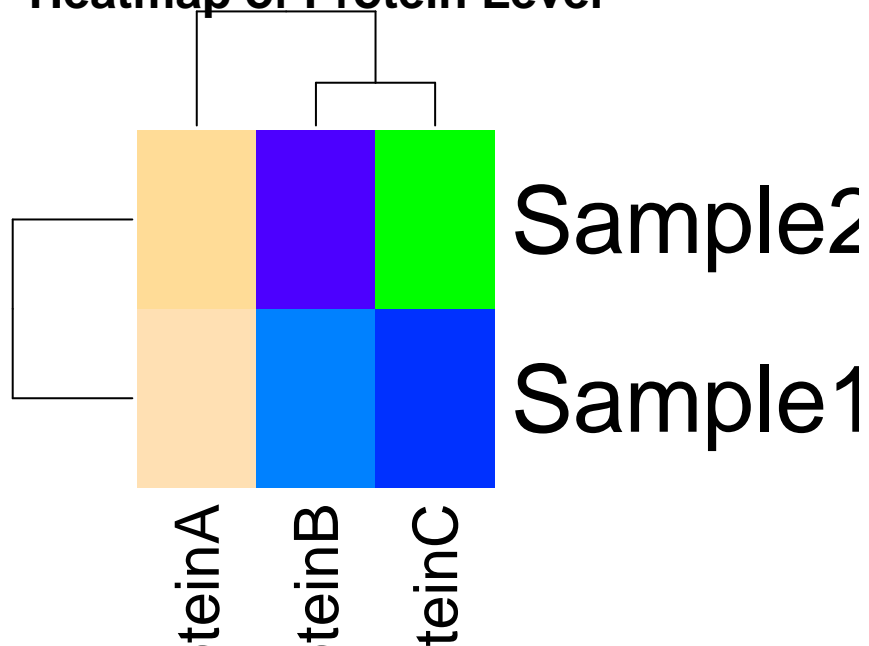
- 1. MARplot


```
# generating MARplot-style scatter plot
plot(GeneExpression, Protein_matrix, type="p", main="Protein level vs. Gene Expression level",
labels <- "Sample-Gene"
text(GeneExpression, Protein_matrix, labels = labels, pos=3)
```



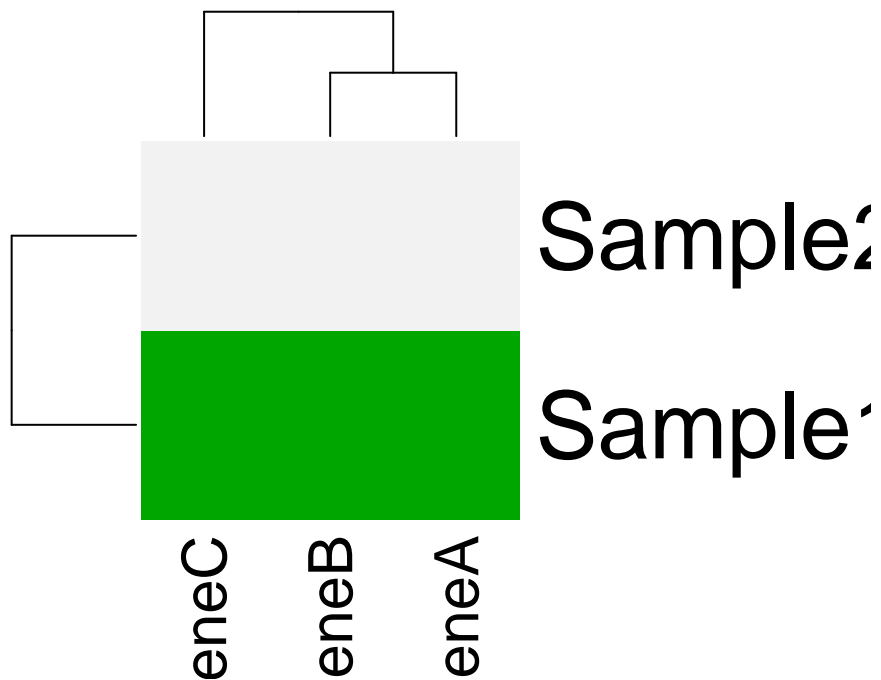
```
# generating a heatmap
heatmap(Protein_matrix, main= "Heatmap of Protein Level", Rowv = TRUE, Colv = TRUE, labRowv = TRUE, labColv = TRUE)
```

Heatmap of Protein Level



- 2. Heatmap of Expression:

```
heatmap(GeneExpression, col = terrain.colors(10), scale = "column")
```



1.2.6 Interpretation

1. Matrix multiplication allows each gene in both samples to be multiplied to their respective translation efficiency. So, the product shows how successfully each gene is translated”)
2. The diagonal TranslationMatrix make sense biologically because they show translation efficiency of each gene and there is no other interaction between them. Although there could be interaction in real-world scenarios.
3. If Sample2 has higher protein levels even with similar gene expression, it means that more mRNAs are translated to proteins compared to Sample1”
4. The upward trend in MARplot may indicate an increase in translation efficacy and downward trend may indicate a decline in translation efficacy”
5. Clustering in the heatmap may suggest which samples are most similar to each other based on their prot.

1.3 Task 3: Animal Breeding – Bull Ranking by Economic Traits

1.3.1 Define Data

```
# Define Bull EBVs
BulleEBVs <- matrix(c(
  400, 1.2, 0.8,
  500, 1.5, 0.6
), nrow = 2, byrow = TRUE)

rownames(BulleEBVs) <- c("Bull1", "Bull2")
colnames(BulleEBVs) <- c("Milk_yield", "Growth_rate", "Fertility")
BulleEBVs
```

	Milk_yield	Growth_rate	Fertility
Bull1	400	1.2	0.8
Bull2	500	1.5	0.6

```
# Define Economic Weights
EconomicWeights <- matrix(c(0.002, 50, 100), ncol = 1)
rownames(EconomicWeights) <- colnames(BulleEBVs)
colnames(EconomicWeights) <- c("Weight")
EconomicWeights
```

	Weight
Milk_yield	2e-03
Growth_rate	5e+01
Fertility	1e+02

1.3.2 Compute Total Economic Value

```
TotalValue <- BullEBVs %*% EconomicWeights
colnames(TotalValue) <- c("Merit")
TotalValue
```

	Merit
Bull1	140.8
Bull2	136.0

Interpretation

Bull1: $(400 \times 0.002) + (1.2 \times 50) + (0.8 \times 100) = 140.8$

Bull2: $(500 \times 0.002) + (1.5 \times 50) + (0.6 \times 100) = 136.0$

Bull1 is more valuable economically.

Biological Interpretation

Economic weights convert genetic merit (EBVs, Estimated Breeding Values) into economic merit. Traits with higher financial importance have a larger impact, regardless of absolute EBV values.

1.3.3 Multiply with Identity Matrix

```
I3 <- diag(3)
rownames(I3) <- colnames(BullEBVs)
colnames(I3) <- colnames(BullEBVs)
I3
```

	Milk_yield	Growth_rate	Fertility
Milk_yield	1	0	0
Growth_rate	0	1	0
Fertility	0	0	1

```
BulleEBVs_identity <- BulleEBVs %*% I3
BulleEBVs_identity
```

	Milk_yield	Growth_rate	Fertility
Bull1	400	1.2	0.8
Bull2	500	1.5	0.6

Interpretation

Multiplying by identity matrix returns the original matrix. It confirms that EBV structure is preserved.

1.3.4 Remove Milk Yield and Recalculate Total Value

```
BulleEBVs_noMilk <- BulleEBVs[, -1]
EconomicWeights_noMilk <- EconomicWeights[2:3, , drop = FALSE]

TotalValue_noMilk <- BulleEBVs_noMilk %*% EconomicWeights_noMilk
colnames(TotalValue_noMilk) <- c("New_Merit")
TotalValue_noMilk
```

	New_Merit
Bull1	140
Bull2	135

Interpretation

Bull1: $(1.2 \times 50) + (0.8 \times 100) = 140$

Bull2: $(1.5 \times 50) + (0.6 \times 100) = 135$

Bull1 still ranks higher, but by a smaller margin.

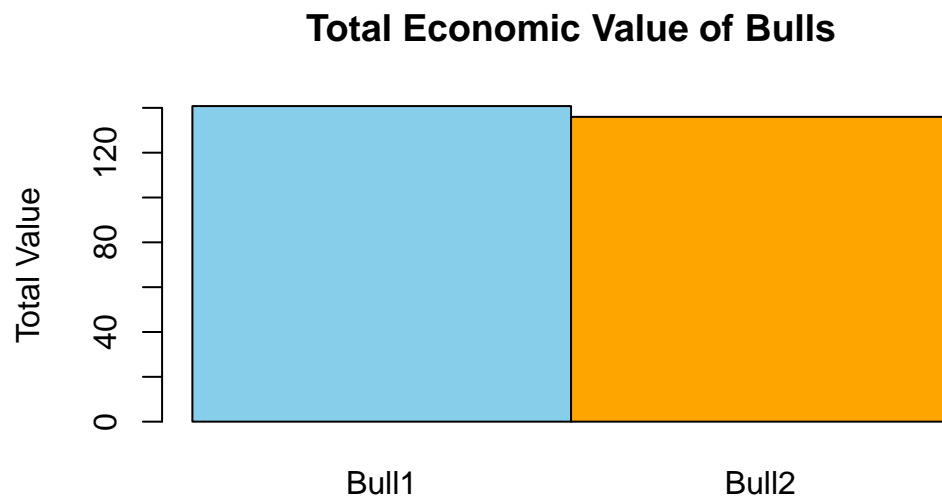
1.3.5 Bar Plot: Total Economic Value

```
barplot(
  TotalValue,
  beside = TRUE,
  names.arg = rownames(BulleEBVs),
  col = c("skyblue", "orange"),
```

```

main = "Total Economic Value of Bulls",
ylab = "Total Value"
)

```

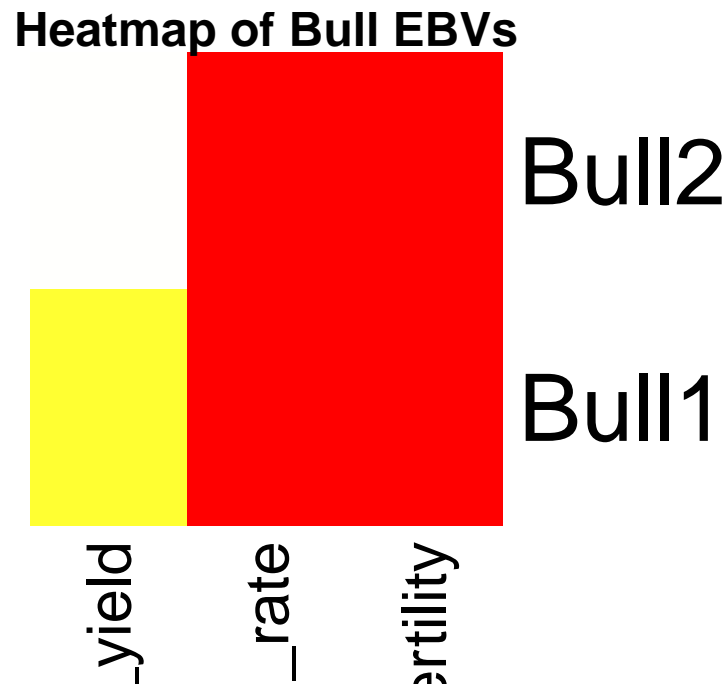


1.3.6 Heatmap of EBVs

```

heatmap(
  BullEBVs,
  Rowv = NA,
  Colv = NA,
  scale = "none",
  col = heat.colors(256),
  main = "Heatmap of Bull EBVs"
)

```



1.3.7 Interpretation Questions

- How do economic weights affect trait importance?

Traits with higher weights contribute more to the total economic value. This makes them more influential in ranking and selection.

- Why might you ignore milk yield?

Milk yield may be excluded in systems focusing on fertility, growth, or when it is no longer a limiting factor. Environmental or economic contexts may also shift trait priorities.

- What is the value of heatmaps?

Heatmaps visually compare EBVs across bulls and traits. They help detect patterns, outliers, and clusters easily in multivariate data.

- Can this method be extended to more bulls and traits?

Yes. This method scales to any number of bulls or traits. Just ensure the EBVs matrix and economic weights are dimensionally compatible.

1.4 Task 4: Plant Breeding – Trait Contributions from Parental Lines

1.4.1 Define Data

Parent trait values (normalized 1–10)

```
ParentTraits <- matrix(c(
  7, 5, 3,
  6, 8, 4,
  5, 6, 6
), nrow = 3, byrow = TRUE)

rownames(ParentTraits) <- c("P1", "P2", "P3")
colnames(ParentTraits) <- c("Drought_resistance", "Yield", "Maturation_time")
ParentTraits
```

	Drought_resistance	Yield	Maturation_time
P1	7	5	3
P2	6	8	4
P3	5	6	6

```
# Define Hybrid Weights
HybridWeights <- matrix(c(0.5, 0.3, 0.2), nrow = 1)
colnames(HybridWeights) <- colnames(ParentTraits)
rownames(HybridWeights) <- c("Weight")
HybridWeights
```

	Drought_resistance	Yield	Maturation_time
Weight	0.5	0.3	0.2

1.4.2 Compute HybridTraits Vector

```
HybridTraits <- HybridWeights %*% ParentTraits
rownames(HybridTraits) <- c("Contribution")
colnames(HybridTraits) <- rownames(ParentTraits)
HybridTraits
```

	P1	P2	P3
Contribution	6.3	6.1	3.9

Interpretation

$$\text{HybridTraits} = (0.5 \times P1) + (0.3 \times P2) + (0.2 \times P3)$$

$$\text{Drought_resistance} = (0.5 \times 7) + (0.3 \times 6) + (0.2 \times 5) = 6.3$$

$$\text{Yield} = (0.5 \times 5) + (0.3 \times 8) + (0.2 \times 6) = 6.1$$

$$\text{Maturation_time} = (0.5 \times 3) + (0.3 \times 4) + (0.2 \times 6) = 3.9$$

The hybrid is moderately strong in drought resistance and yield, and has a relatively shorter maturation time.

Biological Meaning of Unequal Contribution

When one parent contributes more to a trait, it suggests that the trait's heritable strength comes disproportionately from that parent. Breeders can use this knowledge to amplify desirable traits using the best parent.

1.4.3 Multiply with Identity Matrix

```
I3 <- diag(3)
ParentTraits_identity <- ParentTraits %*% I3
colnames(ParentTraits_identity) <- colnames(ParentTraits)
ParentTraits_identity
```

	Drought_resistance	Yield	Maturation_time
P1	7	5	3
P2	6	8	4
P3	5	6	6

Interpretation

Multiplying by identity matrix returns the original matrix. This operation verifies structural consistency and dimensionality.

1.4.4 Remove T3 (Maturation Time) and Recalculate

```
ParentTraits_T1T2 <- ParentTraits[, 1:2]
ParentTraits_T1T2
```

	Drought_resistance	Yield
P1	7	5
P2	6	8
P3	5	6

```
HybridTraits_T1T2 <- HybridWeights %*% ParentTraits_T1T2
HybridTraits_T1T2
```

	Drought_resistance	Yield
Weight	6.3	6.1

Interpretation

Drought_resistance = $(0.5 \times 7) + (0.3 \times 6) + (0.2 \times 5) = 6.3$

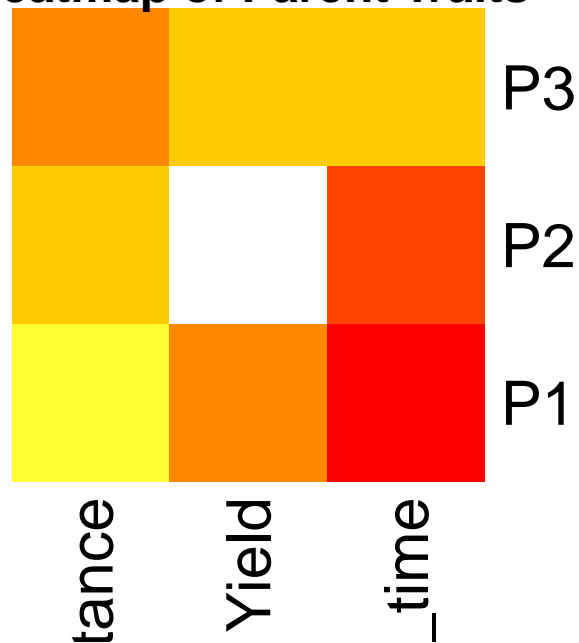
Yield = $(0.5 \times 5) + (0.3 \times 8) + (0.2 \times 6) = 6.1$

Removing a trait (T3) changes the trait profile. Hybrid selection may now favor traits that remain.

1.4.5 Heatmap of ParentTraits

```
heatmap(
  ParentTraits,
  Rowv = NA,
  Colv = NA,
  scale = "none",
  col = heat.colors(256),
  main = "Heatmap of Parent Traits"
)
```

Heatmap of Parent Traits



1.4.6 Bar Plot of Hybrid Traits

```
barplot(  
  HybridTraits,  
  beside = TRUE,  
  names.arg = colnames(ParentTraits),  
  col = c("#66c2a5", "#fc8d62", "#8da0cb"),  
  main = "Hybrid Trait Profile",  
  ylab = "Trait Value"  
)
```

Hybrid Trait Profile



1.4.7 Interpretation Questions

- How does the weighting of parents affect the hybrid's performance?

Stronger weights mean more genetic contribution. Traits from highly weighted parents dominate the hybrid profile.

- What does the identity matrix represent here?

It represents a neutral transformation. It confirms data integrity when used in matrix multiplication.

- If you used equal weights (for each), how would the hybrid traits change?

Traits would reflect an even mix, potentially leading to balanced but less specialized performance.

- What real-world limitations does this simplified model ignore?

- i. Non-additive genetic effects (dominance, epistasis)
- ii. Environmental interactions
- iii. Trait heritability and correlations
- iv. Breeding feasibility and cost

1.5 Task 5: Managing Matrices and Weight Vectors Using Lists in R

Now that we've explored trait-based decisions using matrices, it's time to organize our work using R's list structure. Lists help bundle related objects like matrices and weight vectors, keeping the analysis modular and scalable.

1.5.1 Create a Master List

```
# Assuming previous matrices and weights are already defined:

# Making a MasterList
bioList = list(
  ProteinConc = list(matrix = ProteinMatrix, weights = WeightVector),
  ProteinMap = list(matrix = GeneExpression, weights = TranslationMatrix),
  Animal = list(matrix = BulleEBVs, weights = EconomicWeights),
  Plant = list(matrix = ParentTraits, weights = HybridWeights)
)

print(bioList)
```

```
$ProteinConc
$ProteinConc$matrix
      ProteinX ProteinY ProteinZ
Sample1      5      3      2
Sample2      7      6      4
```

```
$ProteinConc$weights
      Weight
ProteinX  0.5
ProteinY  1.0
ProteinZ  1.5
```

```
$ProteinMap
$ProteinMap$matrix
      GeneA GeneB GeneC
Sample1   10    8    5
Sample2   15   12   10
```

```
$ProteinMap$weights
      protA protB protC
```

GeneA	1.5	0.0	0.0
GeneB	0.0	1.2	0.0
GeneC	0.0	0.0	1.8

```
$Animal
$Animal$matrix
      Milk_yield Growth_rate Fertility
Bull1      400          1.2      0.8
Bull2      500          1.5      0.6
```

```
$Animal$weights
      Weight
Milk_yield 2e-03
Growth_rate 5e+01
Fertility   1e+02
```

```
$Plant
$Plant$matrix
      Drought_resistance Yield Maturation_time
P1              7      5              3
P2              6      8              4
P3              5      6              6
```

```
$Plant$weights
      Drought_resistance Yield Maturation_time
Weight              0.5   0.3              0.2
```

1.5.2 List the Structure

```
names(bioList)          # Top-level list names
```

```
[1] "ProteinConc" "ProteinMap" "Animal"      "Plant"
```

```
lengths(bioList)        # Number of components in each sublist
```

ProteinConc	ProteinMap	Animal	Plant
2	2	2	2

Interpretation

Each top-level entry (e.g., ProteinConc, Plant) contains two components:

- A matrix (e.g., ProteinMatrix)
- A corresponding weight vector or matrix

1.5.3 Indexing Elements from Lists

```
# Access the trait matrix for Plant
bioList$Plant[[1]]
```

	Drought_resistance	Yield	Maturation_time
P1	7	5	3
P2	6	8	4
P3	5	6	6

```
#or
bioList$Plant$matrix
```

	Drought_resistance	Yield	Maturation_time
P1	7	5	3
P2	6	8	4
P3	5	6	6

```
# Access the weight vector for ProteinConc
bioList$ProteinConc[[2]]
```

	Weight
ProteinX	0.5
ProteinY	1.0
ProteinZ	1.5

```
#or
bioList$ProteinConc$weights
```

	Weight
ProteinX	0.5
ProteinY	1.0
ProteinZ	1.5

Interpretation

Use double brackets `[[]]` to extract unnamed list elements by position. But we named our list, so they are easily extractable using the `$` notation.

1.5.4 Perform Weighted Calculations

```
# Protein concentration score
bioList$ProteinConc$matrix %*% bioList$ProteinConc$weights
```

	Weight
Sample1	8.5
Sample2	15.5

```
# Gene → Protein contribution
bioList$ProteinMap$matrix %*% bioList$ProteinMap$weights
```

	protA	protB	protC
Sample1	15.0	9.6	9
Sample2	22.5	14.4	18

```
# Bull economic value
bioList$Animal$matrix %*% bioList$Animal$weights
```

	Weight
Bull1	140.8
Bull2	136.0

```
# Hybrid trait value
bioList$Plant$weights %*% bioList$Plant$matrix
```

	Drought_resistance	Yield	Maturation_time
Weight	6.3	6.1	3.9

1.5.5 Subset and Recalculate

```
# Remove last trait from ParentTraits
ParentSubset <- bioList$Plant$matrix[, 1:2]
NewWeights <- matrix(c(0.6, 0.4), nrow = 2)

# Recalculated hybrid score
SubsetHybridScore <- ParentSubset %*% NewWeights
SubsetHybridScore
```

```
      [,1]
P1    6.2
P2    6.8
P3    5.4
```

Interpretation

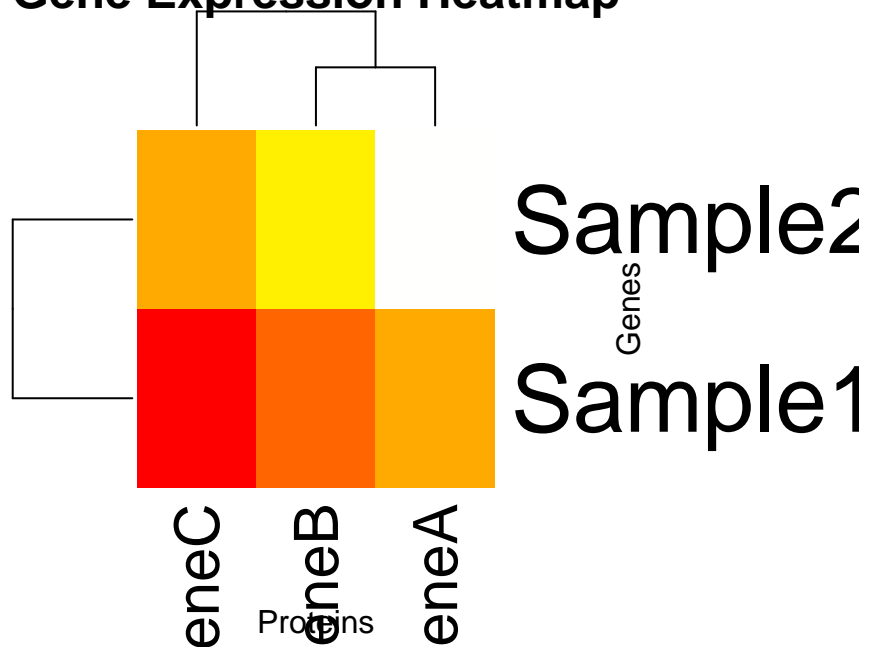
Dropping a trait and reweighting highlights its influence in trait aggregation and selection.

1.5.6 Visualization Tasks

1.5.6.1 Heatmap: Gene Expression

```
heatmap(
  bioList$ProteinMap$matrix,
  scale = "none",
  col = heat.colors(256),
  main = "Gene Expression Heatmap",
  xlab = "Proteins",
  ylab = "Genes"
)
```

Gene Expression Heatmap



1.5.6.2 Bar Plots

1.5.6.2.1 Hybrid traits

```
barplot(  
  bioList$Plant$weights %*% bioList$Plant$matrix,  
  beside = TRUE,  
  main = "Hybrid Trait Contributions",  
  col = "#66c2a5",  
  ylab = "Score"  
)
```

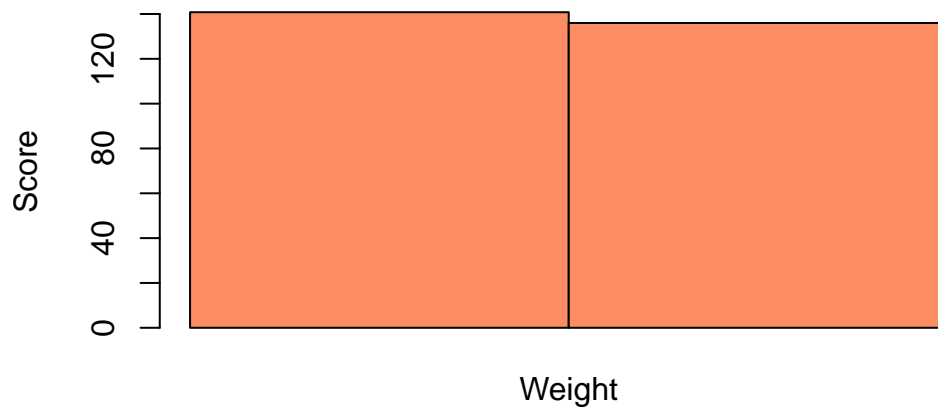
Hybrid Trait Contributions



1.5.6.2.2 Bull EBV (Economic Breeding Values)

```
barplot(  
  bioList$Animal$matrix %*% bioList$Animal$weights,  
  beside = TRUE,  
  main = "Bull EBVs (Economic Values)",  
  col = "#fc8d62",  
  ylab = "Score"  
)
```

Bull EBVs (Economic Values)



1.5.7 Interpretation Questions

- Why use a list structure?

Keeps each dataset and its weights together. Facilitates automated workflows and reuse.

- What's tricky about `[[]]` access?

You must remember the order (`[[1]] = matrix`, `[[2]] = weights`). No names means you can't use `$matrix`, only positional access.

Loop across all list entries

Weighted scores for all entries `lapply(bioList, function(x) x[[2]] %*% x[[1]])`

- How does this help in large-scale pipelines?

You can use this format with `lapply()`, `purrr::map()`, or in targets pipelines for reproducibility and modular processing.

1.6 Homework solutions: Factors, Subsetting, and Biological Insight

1. **Character vs Factor** A character vector simply holds string values, but a factor is a categorical variable with fixed levels, used especially in modeling.

For `mutation_status`, a factor ensures consistent categories (e.g., "Yes" or "No") and helps control level order and statistical reference groups.

2. Factor Levels

```
species <- c("Lactobacillus", "Bacteroides", "Escherichia", "Bacteroides", "Lactobacillus")
species_factor <- factor(species, levels = c("Bacteroides", "Escherichia", "Lactobacillus"))
levels(species_factor)
```

```
[1] "Bacteroides" "Escherichia" "Lactobacillus"
```

Because we defined the level order explicitly, R maintains that order regardless of data input.

3. Ordered Factor Comparison

```
disease_severity <- factor(c("Mild", "Severe", "Moderate"), levels = c("Mild", "Moderate", "Severe"))
disease_severity[1] < disease_severity[2]
```

```
[1] TRUE
```

```
# TRUE
```

“Mild” is less severe than “Severe” based on the defined order.

4. Proportion Extraction

```
prop <- prop.table(table(species_factor))  
prop["Escherichia"]
```

```
Escherichia  
0.2
```

`prop$Escherichia` won't work — named numeric vectors require bracket-based access.

5. Subsetting by Conditions

```
gene_df <- data.frame(  
  gene_id = c("BRCA1", "TP53", "MYC", "EGFR", "GAPDH"),  
  expression = c(8.2, 6.1, 9.5, 7.0, 10.0),  
  mutation = factor(c("Yes", "No", "Yes", "No", "No")),  
  pathway = c("DNA Repair", "Apoptosis", "Cell Cycle", "Signaling", "Metabolism")  
)  
rownames(gene_df) <- gene_df$gene_id #name the rows by the gene IDs  
gene_df <- gene_df[, -1] #remove the first column which is not needed anymore  
#gene_df  
gene_df[gene_df$expression > 7 & gene_df$mutation == "No", ]
```

```
      expression mutation    pathway  
GAPDH          10      No Metabolism
```

Returns genes with **high expression** (>7) and **no mutation** — potentially highly active but wild-type genes.

6. Group-wise Expression Summary

The given vectors are:

```
samples <- c("WT", "KO", "WT", "KO", "WT")  
expression <- c(5.2, 8.1, 4.3, 9.0, 5.7)
```

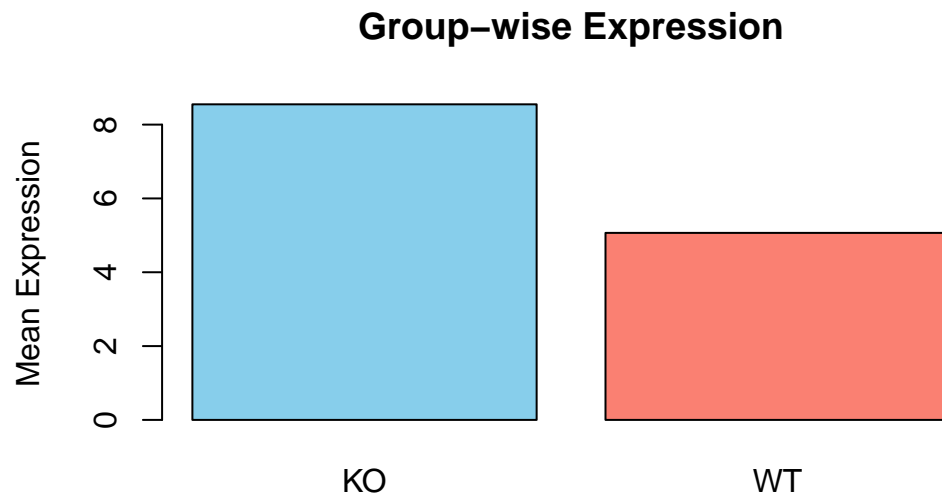
The solution would be:

```
group_factor <- factor(samples)

# Mean expression
tapply(expression, group_factor, mean) ## KO: 8.55, WT: 5.07
```

KO	WT
8.550000	5.066667

```
# Plot
barplot(tapply(expression, group_factor, mean),
        col = c("skyblue", "salmon"),
        ylab = "Mean Expression",
        main = "Group-wise Expression")
```



7. Gene Subsetting

```
gene_df[gene_df$expression > 8 &
        gene_df$pathway %in% c("Cell Cycle", "Signaling"), ]
```

	expression	mutation	pathway
MYC	9.5	Yes	Cell Cycle

It filters for genes highly expressed and involved in key biological pathways.

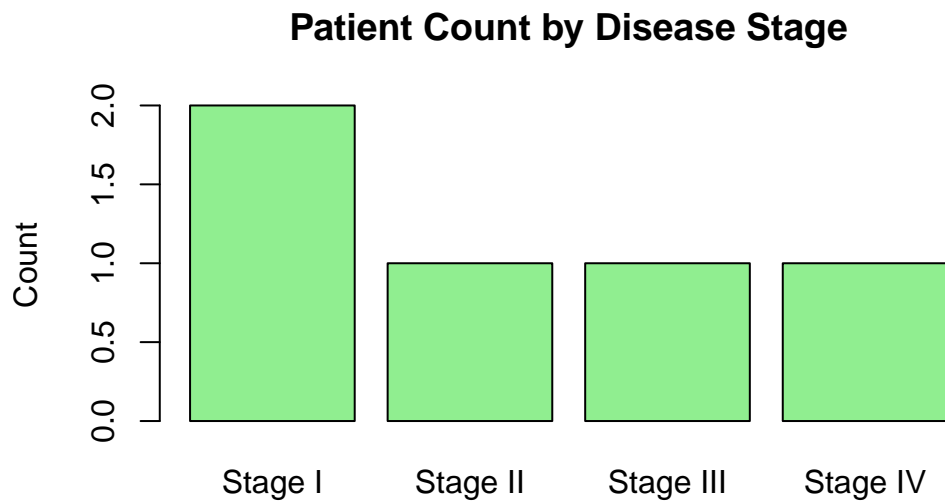
8. Disease Stage Visualization

```

stages <- c("Stage I", "Stage III", "Stage II", "Stage IV", "Stage I")
disease_stage <- factor(stages,
                        levels = c("Stage I", "Stage II", "Stage III", "Stage IV"),
                        ordered = TRUE)

barplot(table(disease_stage),
        col = "lightgreen",
        main = "Patient Count by Disease Stage",
        ylab = "Count")

```



Let's do the severity order check:

```

# Comparison
disease_stage[2] > disease_stage[1] # TRUE

```

```
[1] TRUE
```

So, “Stage III” is more severe than “Stage I”.

9. Oncogene Subsetting and Releveling

```

# Define a small gene dataset
gene_data <- data.frame(
  gene = c("TP53", "BRCA1", "MYC", "GAPDH", "EGFR"),
  expression = c(9.1, 7.3, 10.5, 5.2, 8.6),
  type = factor(c("Tumor Suppressor", "Oncogene", "Oncogene", "Housekeeping", "Oncogene"))
)

```

```
)
```

```
# Subset: Oncogene rows with expression > 8
gene_data[gene_data$type == "Oncogene" & gene_data$expression > 8, ]
```

```
gene expression      type
3 MYC                10.5 Oncogene
5 EGFR                8.6 Oncogene
```

Let's relevel now, "Housekeeping" is the reference:

```
# Relevel: make "Housekeeping" the reference level
gene_data$type <- relevel(gene_data$type, ref = "Housekeeping")

# Check the new levels
levels(gene_data$type)
```

```
[1] "Housekeeping"      "Oncogene"          "Tumor Suppressor"
```

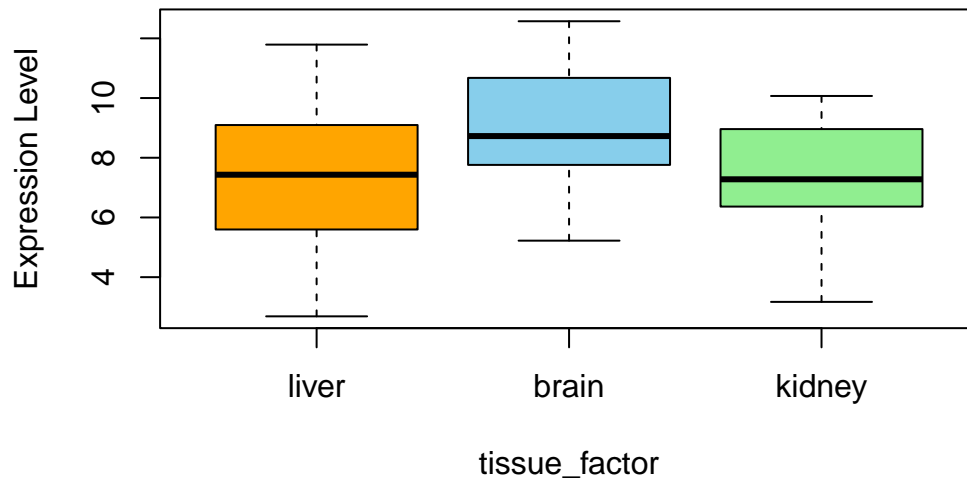
But the "Housekeeping" was already a reference by default (using alphabetic ordering by R). Making something else the reference would make more sense. Our code above work, but nothing new is done.

10. Simulated Expression by Tissue

```
set.seed(42)
gene_expr <- rnorm(45, mean = 8, sd = 2)
tissue <- rep(c("brain", "liver", "kidney"), each = 15)
tissue_factor <- factor(tissue, levels = c("liver", "brain", "kidney"))

boxplot(gene_expr ~ tissue_factor,
        col = c("orange", "skyblue", "lightgreen"),
        main = "Expression by Tissue",
        ylab = "Expression Level")
```


Expression by Tissue



Let's calculate variability per tissue type now:

```
# Variability
tapply(gene_expr, tissue_factor, sd)
```

```
liver    brain    kidney
2.713940 2.050487 1.993668
```

```
# Returns standard deviation per tissue group
```

1.7 Homework Solution tidyverse

```
# Load necessary package
library(ggplot2)

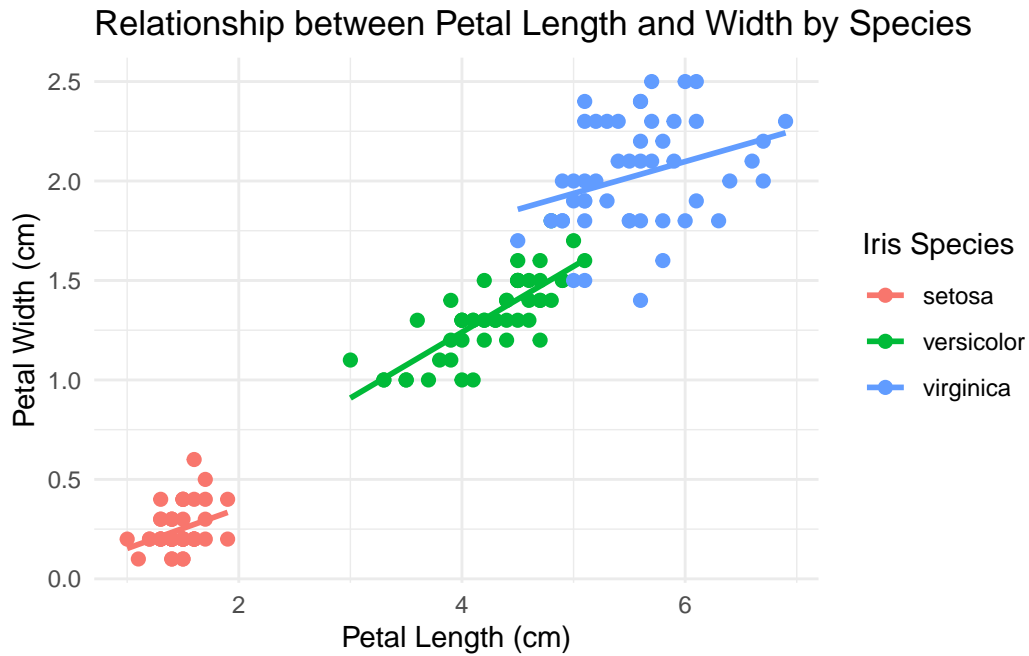
# Use the built-in iris dataset
data(iris)

# Create the plot
ggplot(iris, aes(x = Petal.Length, y = Petal.Width, color = Species)) +
  geom_point(size = 2) + # scatter plot points
  geom_smooth(method = "lm", se = FALSE) + # linear regression lines
  labs(
```

```

title = "Relationship between Petal Length and Width by Species",
x = "Petal Length (cm)",
y = "Petal Width (cm)",
color = "Iris Species"
) +
theme_minimal() # apply a clean minimal theme

```



```

library(ggplot2)

ggplot(iris, aes(x = Petal.Length, y = Petal.Width)) +
  geom_point(aes(color = Species), size = 2) + # color points by Species
  geom_smooth(method = "lm", se = FALSE, color = "black") + # single regression line
  labs(
    title = "Overall Regression: Petal Length vs Width (Iris Dataset)",
    x = "Petal Length (cm)",
    y = "Petal Width (cm)",
    color = "Iris Species"
  ) +
  theme_minimal()

```

Overall Regression: Petal Length vs Width (Iris Dataset)

