Cheat Sheet: ANOVA, Kruskal-Wallis, Chi-Square, and Fisher's Exact Test in R

Preliminary Steps (Before Choosing a Test)

1. Set Working Directory & Import Data

Before you can start your analysis, you need to import your data into R:

```
setwd("path_to_your_directory")  # Set the location where your data file is stored
        or 'Session'→ 'Set Working Directory' → 'Choose Directory'
data <- read.csv("your_file.csv")  # Import your data from a CSV file
```

2. Check Data Normality

To decide which statistical test to use, check if your data is normally distributed.

- **Histogram**: Visualise the distribution of your data.

```
hist(data$variable)  # Replace 'variable' with the name of the column you want to check
```

- **Shapiro-Wilk Test**: Statistically test for normality.

```
shapiro.test(data$variable)  # Replace 'variable' with your column name
```

  - **p > 0.05**: Data is normal (use parametric tests like ANOVA).
  - **p ≤ 0.05**: Data is not normal (use non-parametric tests like Kruskal-Wallis).

---

1. Kruskal-Wallis Test (Non-Parametric)

Use the Kruskal-Wallis test when your data is **not normally distributed** and you want to compare **medians** across multiple groups.

Example: Comparing weights across months.

Code:

```
kruskal.test(weight ~ month, data = dataset)
```

- weight: Continuous variable (e.g., weight of animals).
- month: Categorical variable (e.g., different months).

Interpreting the Results:

- **p ≤ 0.05**: Significant difference between groups.
- **p > 0.05**: No significant difference between groups.

Visualising Results:

You can create a **box plot** to show the distribution of data across groups:

```
library(ggplot2)
ggplot(dataset, aes(x = month, y = weight)) +
  geom_boxplot(fill = "green", color = "red") +
  labs(x = "Month", y = "Weight (g)") +
  theme_classic()
```

---

2. ANOVA (Analysis of Variance)

ANOVA is used when your data is **normally distributed (parametric),** and you want to compare the **means** across multiple groups.

Steps:

1. **Run ANOVA**:

```
anova_result <- aov(weight ~ month, data = dataset)
summary(anova_result)
```

   - weight: Continuous variable.
   - month: Categorical variable.

Interpreting the Results:

- **p ≤ 0.05**: There is a significant difference between the group means.
- **p > 0.05**: No significant difference between the group means.

Check Residuals:

Residuals show the difference between the observed and predicted values. It's important to check if these are normally distributed:

```
hist(resid(anova_result))  # Visualise residuals
shapiro.test(resid(anova_result))  # Test normality of residuals
```

Visualising Results:

You can create a **bar chart with error bars** to show the means and their variability:

```
ggplot(data = summary_table, aes(x = month, y = mean_weight)) +
  geom_bar(stat = "identity", fill = "blue") +
  geom_errorbar(aes(ymin = mean_weight - sd_weight, ymax = mean_weight +
sd_weight), width = 0.2) +
  labs(x = "Month", y = "Mean Weight (g)") +
  theme_classic()
```

---

3. Chi-Square Test, (Goodness-of-fit)

The Chi-Square test compares **observed frequencies** with **expected frequencies** in categorical data.

Steps:

1. **Create a Frequency Table**:

```
table <- table(dataset$category1, dataset$category2)
```

2. **Run Chi-Square Test**:

```
chisq.test(table)
```

Interpreting the Results:

- **p ≤ 0.05**: Significant difference between observed and expected frequencies.
- **p > 0.05**: No significant difference between observed and expected frequencies.

---

4. Fisher's Exact Test

Fisher's Exact Test is used to determine if two categorical variables are significantly associated, especially in **small sample sizes** or when you have **small values** in contingency tables (e.g., 2x2 tables).

Steps:

1. **Create a Contingency Table**:

```
table <- table(dataset$category1, dataset$category2)
```

2. **Run Fisher's Exact Test**:

```
fisher.test(table)
```

Interpreting the Results:

- **p ≤ 0.05**: Significant relationship between the two categories.
- **p > 0.05**: No significant relationship between the two categories.

Error Troubleshooting

Here are some common mistakes and how to fix them:

- **Forgotten Brackets**: Ensure every opening bracket (has a matching closing bracket).
- **Misspelled Variables**: Double-check that your column names in the dataset match exactly with what's in the code.
- **Missing Libraries**: Install necessary libraries before running the code:

```
install.packages("ggplot2")
install.packages("dplyr")
```

Key Commands Summary

Test	**Command**	**Purpose**
Normality Test	shapiro.test(data$variable)	Check if data is normally distributed.
Kruskal-Wallis Test	kruskal.test(weight ~ group, data = dataset)	Compare medians across groups (non-parametric).
ANOVA	aov(response ~ factor, data = dataset)	Compare means across groups (parametric).
Chi-Square Test	chisq.test(table)	Compare observed vs expected frequencies.
Fisher's Test	fisher.test(table)	Test categorical data with small sample sizes.
Box Plot	geom_boxplot()	Visualise data distribution.
Bar Chart	geom_bar(stat = "identity")	Display means with error bars.

Cheat Sheet: Data Aggregation, Visualisation, and Filtering in R

---

1. Why Aggregate Data?

What is Aggregation?

Aggregation involves summarising your data by applying a function (e.g., mean, sum, standard deviation) across groups within your dataset. For example, you might want to calculate the **average weight** for each month in a dataset.

Why Aggregate Data?

Aggregation is important because it helps you simplify large datasets, making it easier to understand trends and patterns across different groups. It also helps to:

- **Simplify Complex Data**: By looking at averages or other summaries, you can quickly get an idea of what's going on.
- **Highlight Patterns**: Aggregation reveals trends, like how the average weight varies by month.
- **Support Analysis**: Aggregated data often serves as the foundation for visualising data or running statistical tests.

---

2. Aggregating Mean and Standard Deviation by Group

Mean Aggregation

To find the **average weight** for each month, we use the aggregate() function. This groups the data by month and calculates the mean for each group.

```
mean_weight_by_month <- aggregate(weight ~ month, data = dataset, FUN = mean)
```

- **weight**: The variable you're summarising (the value you want to find the average for).
- **month**: The grouping variable (the factor you want to group by).
- **FUN = mean**: Tells R to calculate the mean (average) for each group.

Standard Deviation Aggregation

Similarly, you can calculate the **standard deviation** to understand the variability or spread of the data for each month:

```
sd_weight_by_month <- aggregate(weight ~ month, data = dataset, FUN = sd)
```

- **sd**: Standard deviation function, which measures how spread out the values are.

Why Use Aggregation?

- Aggregating mean and standard deviation helps you **understand the central tendency (mean)** and **variability (standard deviation)** of your data.
- It can also highlight whether any particular months have much higher or lower values, suggesting trends or outliers.

---

3. Installing and Using Libraries

R has a lot of **libraries** (or packages) that make it easier to do certain tasks, like visualising data or manipulating data. Two useful libraries for data manipulation and plotting are ggplot2 and dplyr.

Install and Load Libraries

To install a library, use install.packages(). After installation, load it with library():

```
install.packages("ggplot2")  # Install ggplot2
install.packages("dplyr")  # Install dplyr

library(ggplot2)  # Load ggplot2 for plotting
library(dplyr)  # Load dplyr for data manipulation
```

---

4. Grouping and Summarising Data in R with dplyr

When working with grouped data (e.g., monthly measurements), you can calculate summaries like averages and variability using dplyr. Below is an example to group data by month and calculate the mean and standard deviation for each group.

Code: Summarising Data

```
# Load the required library
library(dplyr)

# Group data by 'month2' and calculate mean and standard deviation of 'weight'
summary_vw <- vw %>%              # Start with the dataset 'vw'
  group_by(month2) %>%            # Group data by 'month2' (e.g., months)
  summarise(
    mean_weight = mean(weight),      # Calculate the average weight for each month
    sd_weight = sd(weight)         # Calculate the variation (standard deviation) of weight
  )

# View the summarised data
print(summary_vw)
```

Explanation: What This Does

1. group_by(month2): Groups your dataset by the column month2 (e.g., January, February). Each group will be summarised separately.
2. summarise(): Creates a new table with calculations for each group:
    - mean(weight): The average value of weights in each group.
    - sd(weight): How spread out the weights are in each group.
3. The result, summary_vw, is a new table where:
    - Each row represents a group (e.g., a month).
    - Columns display calculated summaries like the mean and standard deviation for each group.

When to Use This

- Simplifying Data: You want to summarise large datasets by categories like months, regions, or years.
- Exploring Trends: Calculate averages and variability to spot patterns across groups (e.g., weight differences across months).
- Preparing Visuals: Use the summarised data to create charts (e.g., bar charts or error bars).

Output Example
If your dataset has month2 (months) and weight (values), the output might look like this:

month2	mean_weight	sd_weight
January	45.5	2.3
February	50.1	3.7
March	47.8	2.1

This shows that:

- The average weight in January is 45.5, with a standard deviation of 2.3.
- February has higher variability in weights (3.7).

Key Takeaways

- group_by() organises your data by categories.
- summarise() performs calculations for each group, making large datasets easier to interpret.

---

5. Creating Visuals with ggplot2
Visualisation is key to understanding the data. Once you've aggregated your data, you can plot it to reveal patterns.
Creating a Bar Chart with ggplot2
A **bar chart** shows the **mean weight** for each month. Here's how to make a simple bar chart using ggplot2:

```
library(ggplot2)
ggplot(summary_vw, aes(x = month2, y = weight)) +
  geom_bar(stat = "identity", fill = "grey", color = "black") +
  geom_errorbar(aes(ymin=mean_weight - sd_weight, ymax=mean_weight + sd_weight),
width=0.5) +
  theme_classic() +
  labs(x = "Month", y = " Weight (g)")
```

- **geom_bar(stat = "identity")**: This tells ggplot2 to create a bar chart, where the heights of the bars represent the mean weight for each month.
- **fill = "blue", color = "red"**: Customises the colour of the bars and borders.
- **geom_errorbar():** Adds error bars to show variability, with ymin and ymax defining the range (mean ± standard deviation), width = 0.5 controlling width, and color = "black" setting the bar colour.
- **labs()**: Adds axis labels to make the plot easier to interpret.
- **theme_classic()**: Gives the plot a clean look.

Why Visualise Data?

- **Easier to Interpret**: A plot helps you see trends and outliers clearly, something that raw numbers can't always convey.
- **Better Communication**: When presenting results, visual representations like bar charts or box plots are more effective for communication than tables of data.

Why Visualise with Error Bars?

- **Clarifies Uncertainty**: Error bars show the variability of the data and help communicate how reliable the estimates are.

- **Better Interpretation**: Helps the audience understand the spread of the data, not just the averages.

---

6. Filtering Outliers from Your Dataset

Outliers are extreme values that differ significantly from the rest of the data. They can distort statistical analyses, so you might want to remove them.

Create a Filtered Dataset:

If you suspect that certain data points are outliers, use the subset() function to filter them out:

```
filtered_vw <- subset(vw, weight>25&weight<40)  # Remove rows where weight is over 40 and below 25.
```

Why Remove Outliers?

- **Improves Accuracy**: Outliers can skew the results of statistical tests and models.
- **Better Visualisation**: Removing outliers helps you get a clearer visualisation of the data.

In Kerry's example video (ANOVA and Chi Squared), filtering the village weaver data for the weight column makes the data parametric and normally distributed, so we can then use this in an ANOVA parametric test. This would look like this:

```
anova_test <- aov(filtered_vw$weight, filtered_vw$month)
        #Make sure you're using the new filtered dataset, instead of the old "vw" dataset.
summary(anova_test)
```

Key Commands Summary

Action	**Command**	**Purpose**
Filter Data	subset(dataset, condition)	Filter data based on condition (e.g., weight < 100)
Calculate Mean	aggregate(weight ~ month, data = dataset, FUN = mean)	Calculate the mean of a variable by group (e.g., month)
Calculate Standard Deviation	aggregate(weight ~ month, data = dataset, FUN = sd)	Calculate the standard deviation by group
Create Bar Chart	ggplot(dataset, aes(x = month, y = weight)) + geom_bar(stat = "identity")	Create a bar chart of means by group
Add Error Bars	geom_errorbar(aes(ymin = mean_weight - sd_weight, ymax = mean_weight + sd_weight))	Add error bars to a bar chart
Run Shapiro Test	shapiro.test(filtered_data$weight)	Test for normality
Run ANOVA	aov(weight ~ month, data = filtered_data)	Compare means across groups
Install Library	install.packages("library_name")	Install an R library
Load Library	library(library_name)	Load an installed R library