```r
# A genotype example ----
# install.packages('SKAT')
library(SKAT)
library(dplyr)
library(ggplot2)

# Set up data ----
# Set seed for reproducibility
set.seed(123)

# Generate random genetic data
n_subjects <- 200  # number of subjects
n_variants <- 10  # number of variants

# Simulate phenotypes (1: disease, 0: healthy)
phenotype <- rbinom(n_subjects, 1, 0.5)

# Simulate covariates: gender (binary), age (continuous), PC1 (continuous)
gender <- rbinom(n_subjects, 1, 0.5)
age <- rnorm(n_subjects, 50, 10)
PC1 <- rnorm(n_subjects, 0, 1)

# Simulate genotypes (binary: 0/1/2 copies of minor allele)
genotype <- matrix(rbinom(n_subjects * n_variants, 2, 0.5), nrow = n_subjects)

# Melt the genotype matrix for visualization
genotype_melted <- reshape2::melt(genotype)

# Change column names to "sample" and "variant"
colnames(genotype_melted)[1:2] <- c("sample", "variant")

# Plot the genotype matrix ----
library(scales)
p_GM <- ggplot(genotype_melted, aes(y = sample, x = variant, fill = factor(value))) +
    geom_tile() +
    scale_fill_manual(values = c("grey90", "grey50", "grey0")) +
    theme_classic() +
    labs(
        title = "Genotype matrix",
        fill = "Genotype\nvalue"
    )

# p_GM
# Simulate weights
weights <- c(rep(0.5, n_variants / 2), rep(2, n_variants / 2))

# Melt the genotype matrix for visualization
genotype_weights <- reshape2::melt(weights)
genotype_weights$variant <- rownames(genotype_weights) %>% as.numeric()

# Plot the variant weights ----
p_W <- ggplot(genotype_weights, aes(y = 0, x = variant, fill = factor(value))) +
    geom_tile() +
    scale_fill_manual(values = c("pink", "red")) +
    theme_classic() +
    theme(axis.text.y = element_blank()) +
    labs(
        title = "Genotype weights",
        fill = "Genotype\nweights"
    ) +
    ylab("weight")

# p_W
# Weighted genotype matrix
GW <- genotype %*% diag(sqrt(weights))

# Melt the genotype matrix for visualization
GW_melted <- reshape2::melt(GW)

# Change column names to "sample" and "variant"
colnames(GW_melted)[1:2] <- c("sample", "variant")

# Plot the weighted genotype matrix ----
library(scales)
p_GW <- ggplot(GW_melted, aes(y = sample, x = variant, fill = value)) +
    geom_tile() +
    scale_fill_gradientn(colours = c("white", "blue", "red"),
                         values = rescale(c(0, 1, 2))) +
    theme_classic() +

    labs(
        title = "Genotype weighted matrix",
        fill = "Genotype\nvalue\nweighted"
    )
```

```r
# p_GW
# Centering matrix
C <- diag(n_variants) - 1 / n_variants

# Melt the centering matrix for visualization
C_melted <- reshape2::melt(C)
colnames(C_melted)[1:2] <- c("variant_X", "variant_Y")

# Plot the centering matrix ----
p_C <- ggplot(C_melted, aes(x = variant_X, y = variant_Y, fill = value)) +
    geom_tile() +
    scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                                midpoint = 0, limit = c(-min(C), max(C)), space = "Lab",
                                name="Centering\nvalue") +
    theme_classic() +
    labs(
        title = "Centering matrix"
    )

# p_c

library(gridExtra)
# Combine the plots using gridExtra
p_grid <- grid.arrange(p_GM, p_W, p_GW, p_C, ncol = 1)

ggsave(p_grid, filename = "./images/p_grid_GM_GW_GWM_CM.pdf", width = 5, height = 10)

# Compute kernel
# K <- t(GW) %*% GW

# Compute kernel
K <- GW %*% t(GW)

# Comment set : kernel

# Melt the kernel matrix for visualization
K_melted <- reshape2::melt(K)
colnames(K_melted)[1:2] <- c("sample_X", "sample_Y")

library(RColorBrewer)
# Define the YlOrRd color palette
ylorrd_colors <- brewer.pal(9, "YlOrRd")

# Plot the kernel matrix ----
p_k <- ggplot(K_melted, aes(x = sample_X, y = sample_Y, fill = value)) +
    geom_tile() +
    scale_fill_gradientn(colors = ylorrd_colors, name = "Kernel\nvalue") +
    theme_classic() +
    labs(
        title = "Kernel matrix"
    )

# p_k
ggsave(p_k, filename = "./images/p_kernel_1.pdf", width = 5, height = 4.5)

# Plot the kernel matrix clustered ----
# We could also cluster to see the features of the kernel matrix
# Modify the default color scheme
default_palette <- colorRampPalette(ylorrd_colors)
palette(default_palette)
heatmap(K, symm = TRUE)
heatmap_filepath <- "./images/p_kernel_2.pdf"
pdf(heatmap_filepath)
heatmap(K, symm = TRUE)
dev.off()

# Comment set: prompt

# SKAT test ----
# Continuing from kernel_example.R with manual SKAT analysis
# Required library for p-value calculation
library(CompQuadForm)

# Phenotypic data
Y <- phenotype

# Covariates such as gender, age, PCs
Z <- data.frame(gender, age, PC1)
Z <- data.frame(gender, age)

# Create a data frame to contain the variables
data <- data.frame(Y = Y, Z)

# Linear regression model ----
# Create a linear regression formula to adjust variables
formula.H0 <- Y ~ gender + age + PC1
formula.H0 <- Y ~ gender + age
# Fit the linear regression model
linear_model <- lm(formula.H0, data)

# Calculate residuals
res <- resid(linear_model)

# Calculate residual sum of squares
s2 <- sum(res^2)

# Calculate the design matrix for the linear regression model
X1 <- model.matrix(formula.H0, data)

# Identity matrix
D0 <- diag(length(res))

# Create a dataframe from the matrix and melt it
D0_df <- reshape2::melt(D0)

#  Plot the identity matrix ----
p_D0 <- ggplot(D0_df, aes(x=Var1, y=Var2, fill=value)) +
    geom_tile() +
    theme_classic() +
    labs(title = "Identity Matrix (D0)", x = "Row", y = "Column") +
    scale_fill_gradient(low = "white", high = "steelblue")

ggsave(p_D0, filename = "./images/p_D0.pdf", width = 5, height = 4.5)
```

```r
# Calculate projection matrix for the null hypothesis
P0 <- D0 - X1 %*% solve(t(X1) %*% X1) %*% t(X1)

# Create a dataframe from the matrix and melt it
P0_df <- reshape2::melt(P0)

# Plot the projection matrix for the null hypothesis ----
p_P0 <- ggplot(P0_df, aes(x=Var1, y=Var2, fill=value)) +
    geom_tile() +
    theme_classic() +
    labs(title = "Projection Matrix (P0)", x = "Row", y = "Column") +
    scale_fill_gradient(low = "white", high = "steelblue")

ggsave(p_P0, filename = "./images/p_P0.pdf", width = 5, height = 4.5)


# Plot the first set of values from the projection matrix for the null hypothesis to illustrate more clearly ----
p_P0_first <- P0_df %>% filter(Var2 == 1) %>%
    filter(Var1 > 1) %>%
    ggplot(aes(x=Var1, y=value, fill=value)) +
    geom_bar(stat = "identity") +
    theme_classic() +
    labs(title = "Projection Matrix (P0)", x = "Row", y = "Column") +
    scale_fill_gradient(low = "black", high = "steelblue")

ggsave(p_P0_first, filename = "./images/p_P0_first.pdf", width = 5, height = 4.5)

# Calculate the product of projection and kernel matrices
PKP <- P0 %*% K %*% P0

print(dim(P0))
print(dim(K))

# Create a dataframe from the matrix and melt it
PKP_df <- reshape2::melt(PKP)

# Plot the product of projection and kernel matrices ----
p_PKP <- ggplot(PKP_df, aes(x=Var1, y=Var2, fill=value)) +
    geom_tile() +
    theme_classic() +
    labs(title = "Product of Projection and Kernel Matrices (PKP)", x = "Row", y = "Column") +
    scale_fill_gradient(low = "white", high = "steelblue")

ggsave(p_PKP, filename = "./images/p_PKP.pdf", width = 5, height = 4.5)

# Adjusted score statistic
q <- as.numeric(res %*% K %*% res / s2)

# Eigenvalues calculation ----
# Calculate eigenvalues of PKP - q * P0
ee <- eigen(PKP - q * P0, symmetric = TRUE)

# Filter eigenvalues above the tolerance limit
tol = 1e-10
lambda <- ee$values[abs(ee$values) >= tol]

# Plot the lambda ----
p_lambda <- ggplot(data.frame(lambda = lambda, sample = 1:length(lambda)), aes(x = sample, y = lambda)) +
    geom_point(fill = "black", size = 1, alpha = 0.5) +
    theme_classic() +
    labs(title = "Contributions of Lambda to Test Statistic", x = "Sample", y = "Lambda")

ggsave(p_lambda, filename = "./images/p_lambda.pdf", width = 5, height = 4.5)


# P-value calculation ----
# Use Davies method for p-value calculation
acc = 0.00001
lim = 10000
# p_value <- CompQuadForm::davies(q, lambda, acc = acc, lim = lim)

# KAT.pval
#Compute the tail probability of 1-DF chi-square mixtures
KAT.pval <- function(Q.all, lambda, acc=1e-9,lim=1e6){
    pval = rep(0, length(Q.all))
    i1 = which(is.finite(Q.all))
    for(i in i1){
        tmp = davies(Q.all[i],lambda,acc=acc,lim=lim); pval[i] = tmp$Qq
        if((tmp$ifault>0)|(pval[i]<=0)|(pval[i]>=1)) pval[i] = Sadd.pval(Q.all[i],lambda)
    }
    return(pval)
} # KAT.pval used to compute the p-value, which internally uses davies() but also includes additional checks and
adjustments.

# Calculating the p-value
p.value <- KAT.pval(0, lambda=sort(lambda, decreasing=T), acc = acc, lim = lim)

# Save and print the results
res <- list(p_value = p.value, Q_adj = q)
print(res)
```