EXPD Cheat Sheet

Heath Blackmon

2023-11-02

R Statistics Worksheet

This worksheet will guide you through performing different statistical tests in R, including T-tests (single sample, two-sample, and paired two-sample), Chi-Square test, binomial test, ANOVA, correlation, and simple Generalized Linear Models (GLMs).

Before you begin, make sure you have installed R and optionally RStudio, which is an integrated development environment for R.

T-Tests

Single Sample T-test

Objective: Test if the mean of a sample differs significantly from a known or hypothesized population mean.

```
# Hypothesized population mean
mu <- 5

# Sample data
sample_data <- c(4.5, 5.1, 5.8, 4.9, 5.0, 5.3)

# Perform single sample t-test
t.test(sample_data, mu = mu)
```

```
##
##  One Sample t-test
##
## data:  sample_data
## t = 0.56493, df = 5, p-value = 0.5965
## alternative hypothesis: true mean is not equal to 5
## 95 percent confidence interval:
##  4.644976 5.555024
## sample estimates:
## mean of x
##       5.1
```

Two Sample T-test

Objective: Compare the means of two independent groups.

```r
# Group A data
group_a <- c(2.3, 2.9, 3.1, 2.8, 3.0)

# Group B data
group_b <- c(3.5, 3.8, 3.2, 3.9, 4.2)

# Perform two sample t-test
t.test(group_a, group_b)
```

```
##
##  Welch Two Sample t-test
##
## data:  group_a and group_b
## t = -4.0741, df = 7.6776, p-value = 0.003886
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.4131605 -0.3868395
## sample estimates:
## mean of x mean of y
##      2.82      3.72
```

Paired Two Sample T-test

Objective: Compare the means of repeated measures.

```r
# Before intervention
before <- c(120, 112, 132, 104, 115)

# After intervention
after <- c(122, 118, 136, 108, 120)

# Perform paired two sample t-test
t.test(before, after, paired = TRUE)
```

```
##
##  Paired t-test
##
## data:  before and after
## t = -6.3317, df = 4, p-value = 0.003185
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
##  -6.041685 -2.358315
## sample estimates:
## mean difference
##            -4.2
```

Chi-Square Test

Objective: Test for independence between categorical variables.

```r
# Contingency table
data <- matrix(c(10, 20, 30, 40), nrow = 2)

# Perform Chi-Square test
chisq.test(data)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  data
## X-squared = 0.44643, df = 1, p-value = 0.504
```

Binomial Test

Objective: Test if the probability of success in a Bernoulli experiment differs from a known probability.

```r
# Number of successes
x <- 5

# Number of trials
n <- 20

# Hypothesized probability of success
p <- 0.5

# Perform binomial test
binom.test(x, n, p)
```

```
##
##  Exact binomial test
##
## data:  x and n
## number of successes = 5, number of trials = 20, p-value = 0.04139
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.08657147 0.49104587
## sample estimates:
## probability of success
##                   0.25
```

ANOVA

Objective: Compare the means of three or more groups.

```r
# Data for three groups
group1 <- c(6.2, 5.8, 5.9, 6.1, 6.3)
group2 <- c(6.5, 6.7, 6.8, 6.4, 6.5)
group3 <- c(5.9, 6.1, 6.0, 5.8, 6.0)

# Combine data into a data frame
df <- data.frame(values = c(group1, group2, group3),
```

```r
                group = factor(rep(c('Group1', 'Group2', 'Group3'), each = 5)))

# Perform ANOVA
res.aov <- aov(values ~ group, data = df)
summary(res.aov)
```

```
##             Df Sum Sq Mean Sq F value  Pr(>F)
## group        2  1.108  0.5540   20.02 0.00015 ***
## Residuals   12  0.332  0.0277
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Correlation

Objective: Measure the strength and direction of association between two continuous variables.

```r
# Variable X
x <- c(1, 2, 3, 4, 5)

# Variable Y
y <- c(2, 2.5, 3.5, 4.4, 5.1)

# Perform correlation test
cor.test(x, y)
```

```
##
##  Pearson's product-moment correlation
##
## data:  x and y
## t = 18.265, df = 3, p-value = 0.0003581
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9309095 0.9997201
## sample estimates:
##       cor
## 0.9955338
```

Simple Generalized Linear Models (GLMs)

Linear model with discrete and continuous predictors

Objective: Model the relationship between a continuous response variable and predictor variables that are both discrete and continuous.

```r
# Binary outcome data
outcome <-    c(2, 1, 3, 1, 0, 1, 2, 5, 9, 0, 1, 10, 2, 1, 0, 1, 8, 9)

# Predictor data
predictor1 <- c(1, 3, 5, 2, 4, 0, 1, 3, 7, 2, 7, 9, 4, 6, 2, 5, 7, 8)
predictor2 <- c("A", "A", "B", "A", "A", "A", "B", "A", "B", "A", "A", "B", "A", "A", "A", "A", "B", "B
```

```r
# fit the model
model <- glm(outcome ~ predictor1 + predictor2)
summary(model)
```

```
##
## Call:
## glm(formula = outcome ~ predictor1 + predictor2)
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.4077     0.8774  -0.465  0.64887
## predictor1    0.5100     0.2078   2.454  0.02683 *
## predictor2B   4.0957     1.1437   3.581  0.00273 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 3.761914)
##
##     Null deviance: 203.778  on 17  degrees of freedom
## Residual deviance:  56.429  on 15  degrees of freedom
## AIC: 79.649
##
## Number of Fisher Scoring iterations: 2
```

Binary Logistic Regression

Objective: Model the relationship between a binary response variable and one or more predictor variables.

```r
# Binary outcome data
outcome <-   c(0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1)

# Predictor data
predictor <- c(1, 3, 5, 2, 4, 0, 1, 3, 7, 2, 7, 9, 4, 6, 2, 5, 7, 8)

# Perform binary logistic regression
model <- glm(outcome ~ predictor, family = binomial)
summary(model)
```

```
##
## Call:
## glm(formula = outcome ~ predictor, family = binomial)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -6.7694     3.4887  -1.940   0.0523 .
## predictor     1.6797     0.8548   1.965   0.0494 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 24.9533  on 17  degrees of freedom
```

```
## Residual deviance:  7.8397  on 16  degrees of freedom
## AIC: 11.84
##
## Number of Fisher Scoring iterations: 7
```

Poisson Regression

Objective: Model count data.

```r
# Count outcome data
count_data <- c(0, 2, 3, 1, 4, 13, 2, 10, 3)

# Predictor data
predictor <- c(1, 2, 3, 4, 5, 8, 2, 7, 1)

# Perform Poisson regression
model <- glm(count_data ~ predictor, family = poisson)
summary(model)
```

```
##
## Call:
## glm(formula = count_data ~ predictor, family = poisson)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.13082    0.42557  -0.307    0.759
## predictor    0.33379    0.06858   4.867 1.13e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 33.0918  on 8  degrees of freedom
## Residual deviance:  6.9685  on 7  degrees of freedom
## AIC: 36.018
##
## Number of Fisher Scoring iterations: 5
```

Using the step function to find the best model

Objective: Find the "best" model.

```r
# create the data needed for this example
set.seed(123)
n <- 100
x1 <- rnorm(n)
x2 <- rnorm(n)
x3 <- rnorm(n)
y <- 1 + 2 * x1 + 3 * x2 + .01 * x3 + rnorm(n)
data <- data.frame(y, x1, x2, x3)

initial_model <- glm(y ~ x1 + x2 + x3, data = data)
summary(initial_model)
```

```
##
## Call:
## glm(formula = y ~ x1 + x2 + x3, data = data)
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.98067    0.10734   9.136 1.07e-14 ***
## x1           1.94455    0.11688  16.638  < 2e-16 ***
## x2           3.04622    0.10946  27.831  < 2e-16 ***
## x3          -0.04739    0.11223  -0.422    0.674
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 1.105679)
##
##     Null deviance: 1227.24  on 99  degrees of freedom
## Residual deviance:  106.15  on 96  degrees of freedom
## AIC: 299.75
##
## Number of Fisher Scoring iterations: 2
```

```
best_model <- step(initial_model)
```

```
## Start:  AIC=299.75
## y ~ x1 + x2 + x3
##
##        Df Deviance    AIC
## - x3    1   106.34 297.94
## <none>      106.15 299.75
## - x1    1   412.21 433.42
## - x2    1   962.54 518.23
##
## Step:  AIC=297.94
## y ~ x1 + x2
##
##        Df Deviance    AIC
## <none>      106.34 297.94
## - x1    1   419.52 433.18
## - x2    1   962.61 516.24
```

```
summary(best_model)
```

```
##
## Call:
## glm(formula = y ~ x1 + x2, data = data)
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.9743     0.1058   9.207 6.91e-15 ***
## x1            1.9509     0.1154  16.902  < 2e-16 ***
## x2            3.0451     0.1090  27.947  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for gaussian family taken to be 1.096312)
##
##     Null deviance: 1227.24  on 99  degrees of freedom
## Residual deviance:  106.34  on 97  degrees of freedom
## AIC: 297.94
##
## Number of Fisher Scoring iterations: 2
```

Monte Carlo

Objective: Using simulations to produce a null distribution.

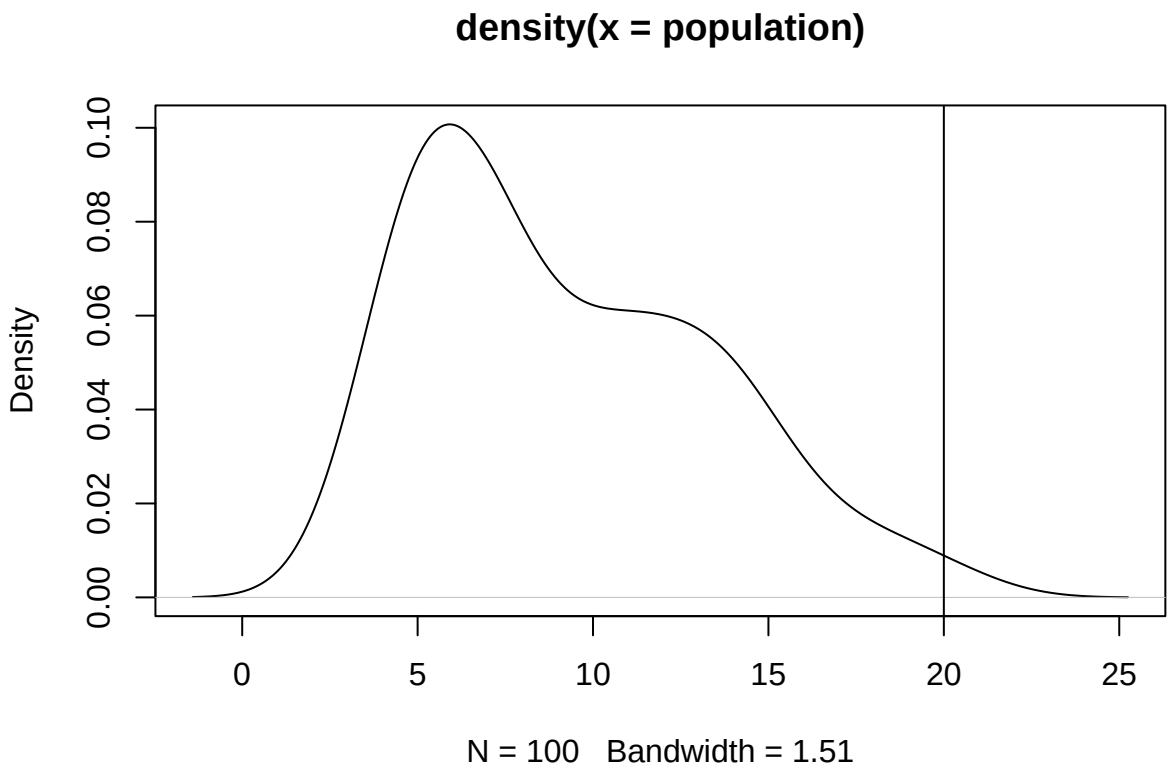
```r
# Biological Example: Bacterial Growth Simulation

# We will simulate the growth of a bacterial population under controlled
# conditions. The growth rate is subject to random fluctuations, but we know
# the average growth rate. We have observed our results which indicated 20
# colonies

observed <- 20
growth_rate <- 1.02  # average growth rate
time_steps <- 100   # number of time steps
n_simulations <- 1000  # number of simulations

result <- c()
set.seed(1)  # for reproducibility
for(i in 1:n_simulations){
  population <- 10
  for (t in 2:time_steps) {
        growth_factor <- rnorm(1, mean = growth_rate, sd = 0.2)
        population[t] <- population[t - 1] * growth_factor
  }
  result[i] <- population[t]
}


plot(density(population))
abline(v=observed)
```




```
pval <- sum(population>=observed)/n_simulations
pval
```

```
## [1] 0.001
```

Permutation Test

Objective: Non-parametric approach for many problems

```
# Simulating two groups of data
group_A <- rnorm(30, mean = 40, sd = 10)
group_B <- rnorm(30, mean = 55, sd = 10)
data <- data.frame(value = c(group_A, group_B),
                   group = rep(c("A", "B"), each = 30))

n_perm <- 1000
observed_diff <- abs(mean(data$value[data$group == "A"]) - mean(data$value[data$group == "B"]))
perm_diffs <- numeric(n_perm)
for (i in 1:n_perm) {
  shuffled <- sample(data$value)
  perm_diffs[i] <- abs(mean(shuffled[data$group == "A"]) - mean(shuffled[data$group == "B"]))
}
p_value <- mean(perm_diffs >= observed_diff)
p_value
```

```
## [1] 0.001
```

When you run these commands in R, replace the placeholder data with your actual dataset. Make sure to install any required packages and load them with `library()` if needed. For complex analyses and larger datasets, it is recommended to use appropriate data structures like data frames and to check the assumptions of each statistical test before interpreting the results.