

R Code to Complement Math Review Packet

A Note:

The notes and code in this document are meant to serve as complementary material to the Math Review Packet. In this program you will be able to perform many calculations in R, R is much more powerful than a calculator. The R code provided is to supplement the math review packet to give you an understanding of how you could use R to solve a variety of problems.

We will cover:

- Logarithms
- Matrices
- Statistical concepts
 - * Central tendency
 - * The Central Limit Theorem
 - * Covariance and correlation
 - * Ordinary Least Square regression

Please use this as a resource to become more comfortable with R. You can run the R chunks by pressing the green play button to run the code to see how one would try to answer the practice questions in R. You can run parts of the code and change portions of it in the Console to better understand what functions do or look up functions in the **Help** tab or type ‘?function’ into the Console.

This is meant to be a tool to provide more practice and context before the start of the program. You can comment (by using ‘#’ in the R chunk) as a way for annotating the code to identify what the code is doing in your own words. This is good practice to understand what code is doing to allow you to write your own code in the future.

Note: You will also see, in this document, LaTeX notation. If you hover over the items in the dollar signs you will see what the formatting will look like when knitted to an pdf.

Properties of Logarithms

In R the `log()` function computed logarithms by natural logarithms ($\ln()$) to calculate logarithms with different bases such as \log_{10} use `log10()`, \log_2 use `log2()`, etc. . .

For example if we wanted to solve $\log_{10} 10000$:

```
log10(10000)
```

```
## [1] 4
```

Additionally if we wanted to solve e^4 , we would use the `exp()`

```
exp(4)
```

```
## [1] 54.59815
```

Practice Problems.

1. Solve the following:

a. $\log_e(e^x) = x$

```
# If x = 3
```

```
log(exp(3))
```

```
## [1] 3
```

b. $\log_{10}(100) = 2$

```
log10(100)
```

```
## [1] 2
```

c. $\log_{10}(\frac{1}{10}) = -1$

```
log10(1/10)
```

```
## [1] -1
```

d. $\log_{10}(0) = \text{No solution}$

```
log10(0)
```

```
## [1] -Inf
```

3 Condense the following c. $\log_{10}(5) + \log_{10}(2) = 1$

```
log10(5) + log10(2)
```

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

Matrix Algebra

Creating a Matrix in R

Type `?matrix` in the **Condole** or `matrix` in **Help** to look at the inputs of the matrix function.

If we want to store the matrix we need give a name to the matrix, “Z” for example, and store the matrix.

For example $\mathbf{Z} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ $\mathbf{Y} = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$

```
#Sorting by row
Z <- matrix(data = c(1,2,3,4,5,6,7,8,9),nrow = 3, ncol = 3, byrow = TRUE)
print(Z)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
```

```
#Sorting by column
Y <- matrix(data = (1:9), nrow = 3, ncol = 3, byrow = FALSE)
print(Y)
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

Definitions

The transpose of an $m \times n$ matrix, A is the $n \times m$ matrix (denoted A^T) such that every element a_{ij} in matrix A is moved to row j and column i . For example, if:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 5 \\ 3 & 4 & 7 \end{bmatrix}$$

then,

$$\mathbf{A}^T = \begin{bmatrix} 1 & 3 \\ 2 & 4 \\ 5 & 7 \end{bmatrix}$$

```
# Matrix A
A <- matrix(data = c(1,2,5,3,4,7), nrow = 2, ncol = 3, byrow = TRUE)
print(A)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    5
## [2,]    3    4    7
```

```
# A transpose
print(t(A))
```

```
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
## [3,]    5    7
```

Identity Matrix

```
# Identity Matrix: use diag() to create an identity matrix
# I = 4x4 identity matrix
I <- diag(4)
print(I)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    0    0    0
## [2,]    0    1    0    0
## [3,]    0    0    1    0
## [4,]    0    0    0    1
```

```
class(I)
```

```
## [1] "matrix"
```

Diagonal

```
X1 <- matrix(data = c(1,0,0,0,4,0,0,0,5), nrow = 3, byrow = TRUE)
print(X1)
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    4    0
## [3,]    0    0    5
```

```
# To obtain the values of the diagonal of an n x n matrix
diag(X1)
```

```
## [1] 1 4 5
```

```
# We could also create an empty matrix and assign values to the diagonal
X2 <- matrix(data = 0, nrow = 3, ncol = 3)
diag(X2) <- c(1,4,5)
print(X2)
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    4    0
## [3,]    0    0    5
```

Upper and Lower Triangular Matrices

```
# Create an empty matrix
V <- W <- matrix(data = 0, nrow = 3, ncol = 3, byrow = TRUE)
print(W)
```

```
##      [,1] [,2] [,3]
## [1,]    0    0    0
## [2,]    0    0    0
## [3,]    0    0    0
```

```
# Use upper.tri() or lower.tri() to change the elements of the upper
# triangular matrix or lower triangular matrix, respectively.
```

```
# Upper Triangular
W[upper.tri(W, diag = TRUE)] <- c(1:6)
print(W)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    4
## [2,]    0    3    5
## [3,]    0    0    6

# Lower Triangular
V[lower.tri(V, diag = TRUE)] <- c(1:6)
print(V)

##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    2    4    0
## [3,]    3    5    6
```

Inverse of a matrix

```
U <- matrix(data = c(1:4), nrow = 2, byrow = TRUE)

# Use solve() to obtain the inverse of a matrix
inU <- solve(U)

# Confirm you get an identity matrix
round(U %*% inU, 2)
```

```
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1

round(inU %*% U, 2)
```

```
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1
```

Determinant of a matrix

```
# Use det() to obtain the determinant of a matrix
det(U)
```

```
## [1] -2
```

Operations with matrices

a. Adding matrices

```
# Using the above matrices (Z and Y) we will show that Z + Y = X
# Adding matrices of the same dimension will produce a matrix of the same dimension
dim(Z + Y); dim(Z); dim(Y)
```

```
## [1] 3 3
## [1] 3 3
## [1] 3 3

Z + Y
```

```
##      [,1] [,2] [,3]
## [1,]    2    6   10
## [2,]    6   10   14
```

```
## [3,] 10 14 18
```

b. Subtracting matrices

```
# Subtracting matrices of the same dimension will produce a matrix of the same dimension  
dim(Z - Y); dim(Z); dim(Y)
```

```
## [1] 3 3
```

```
## [1] 3 3
```

```
## [1] 3 3
```

```
Z - Y
```

```
##      [,1] [,2] [,3]
```

```
## [1,]    0   -2   -4
```

```
## [2,]    2    0   -2
```

```
## [3,]    4    2    0
```

c. Multiplying a matrix by a scalar

```
# Multiplying a matrix by a scalar will produce a matrix of the same dimension  
# Suppose that c = 3
```

```
3 * Z
```

```
##      [,1] [,2] [,3]
```

```
## [1,]    3    6    9
```

```
## [2,]   12   15   18
```

```
## [3,]   21   24   27
```

d. Matrix Multiplication

```
m1 <- matrix(data = c(3,4,2,5,1,2), nrow = 3, ncol = 2, byrow = TRUE)  
print(m1); dim(m1)
```

```
##      [,1] [,2]
```

```
## [1,]    3    4
```

```
## [2,]    2    5
```

```
## [3,]    1    2
```

```
## [1] 3 2
```

```
m2 <- matrix(data = c(7,2,1,3,5,2), nrow = 2, ncol = 3, byrow = TRUE)  
print(m2); dim(m2)
```

```
##      [,1] [,2] [,3]
```

```
## [1,]    7    2    1
```

```
## [2,]    3    5    2
```

```
## [1] 2 3
```

```
# Matrix multiplication
```

```
m1 %*% m2
```

```
##      [,1] [,2] [,3]
```

```
## [1,]   33   26   11
```

```
## [2,]   29   29   12
```

```
## [3,]   13   12    5
```

```
# It is important to note that when multiplying matrices we need to use %*% rather than * because using  
print(Z); print(Y)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```
Z * Y
```

```
##      [,1] [,2] [,3]
## [1,]    1    8   21
## [2,]    8   25   48
## [3,]   21   48   81
```

```
Z %*% Y
```

```
##      [,1] [,2] [,3]
## [1,]   14   32   50
## [2,]   32   77  122
## [3,]   50  122  194
```

Properties of matrix operations

1. $A + B = B + A$

```
Z + Y
```

```
##      [,1] [,2] [,3]
## [1,]    2    6   10
## [2,]    6   10   14
## [3,]   10   14   18
```

```
Y + Z
```

```
##      [,1] [,2] [,3]
## [1,]    2    6   10
## [2,]    6   10   14
## [3,]   10   14   18
```

2. $(A + B) + C = A + (B + C)$

```
(Z + Y) + X1
```

```
##      [,1] [,2] [,3]
## [1,]    3    6   10
## [2,]    6   14   14
## [3,]   10   14   23
```

```
Z + (Y + X1)
```

```
##      [,1] [,2] [,3]
## [1,]    3    6   10
## [2,]    6   14   14
## [3,]   10   14   23
```

3. $(AB)C = A(BC)$

```
(Z %*% Y) %*% X1
```

```
##      [,1] [,2] [,3]
## [1,]   14  128  250
## [2,]   32  308  610
## [3,]   50  488  970
```

```
Z %*% (Y %*% X1)
```

```
##      [,1] [,2] [,3]
## [1,]   14  128  250
## [2,]   32  308  610
## [3,]   50  488  970
```

4. $(A + B)C = AC + BC$

```
(Z + Y) %*% X1
```

```
##      [,1] [,2] [,3]
## [1,]    2   24   50
## [2,]    6   40   70
## [3,]   10   56   90
```

```
(Z %*% X1) + (Y %*% X1)
```

```
##      [,1] [,2] [,3]
## [1,]    2   24   50
## [2,]    6   40   70
## [3,]   10   56   90
```

5. If A is an $m \times n$ matrix, then $I_m A = A$ and $A I_n = A$

```
A
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    5
## [2,]    3    4    7
```

```
Im <- diag(x = 1, nrow = 2)
```

```
In <- diag(x = 1, nrow = 3)
```

```
Im %*% A
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    5
## [2,]    3    4    7
```

```
A %*% In
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    5
## [2,]    3    4    7
```

Note that, in general $AB \neq BA$

```
Z %*% Y
```

```
##      [,1] [,2] [,3]
## [1,]   14   32   50
## [2,]   32   77  122
## [3,]   50  122  194
```

```
Y %*% Z
```



```
##      [,1] [,2] [,3]
## [1,]  66  78  90
## [2,]  78  93 108
## [3,]  90 108 126
```

Writing a system of equations using matrix notation

***** SOPHIE LOOK HERE ***** I don't really know how to show this!

```
a1 <- rep(0, 100)
a2 <- rnorm(100, 0, 1)
a3 <- 2*rnorm(100, 0, 2)
A <- cbind(a1, a2, a3)
dim(A)
```

```
## [1] 100  3
```

```
X <- as.matrix(c(1:3))
dim(X)
```

```
## [1] 3 1
```

```
Y <- A %*% X
dim(Y)
```

```
## [1] 100  1
```

Practice Problems

1. Solve the following:

a.
$$\begin{bmatrix} 2 & 4 & 2 \\ 1 & 3 & 0 \\ 1 & 6 & 2 \end{bmatrix} + \begin{bmatrix} 1 & 5 & 0 \\ -2 & -3 & 0 \\ 1 & 9 & 5 \end{bmatrix} = \begin{bmatrix} 3 & 9 & 2 \\ -1 & 0 & 0 \\ 2 & 15 & 7 \end{bmatrix}$$

```
M1 <- matrix(data = c(2,4,2,1,3,0,1,6,2),
              nrow = 3, ncol = 3, byrow = TRUE)
```

```
M2 <- matrix(data = c(1,5,0,-2,-3,0,1,9,5),
              nrow = 3, ncol = 3, byrow = TRUE)
```

```
M1 + M2
```

```
##      [,1] [,2] [,3]
## [1,]    3    9    2
## [2,]   -1    0    0
## [3,]    2   15    7
```

b.
$$\begin{bmatrix} 2 & 1 \\ -2 & 2 \\ 4 & 2 \end{bmatrix} \begin{bmatrix} 5 & 2 & -1 \\ 3 & 4 & 2 \end{bmatrix} = \begin{bmatrix} 13 & 8 & 0 \\ -4 & 4 & 6 \\ 26 & 16 & 0 \end{bmatrix}$$

```
M3 <- matrix(data = c(2,-2,4,1,2,2), nrow = 3, ncol = 2, byrow = FALSE)
```

```
M4 <- matrix(data = c(5,3,2,4,-1,2), nrow = 2, ncol = 3, byrow = FALSE)
```

```
M3 %*% M4
```

```
##      [,1] [,2] [,3]
## [1,]   13    8    0
## [2,]   -4    4    6
## [3,]   26   16    0
```

c. Let $\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 5 \\ 4 & 0 \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} 4 & 4 \\ 1 & 2 \\ 7 & 0 \end{bmatrix}$ $\mathbf{A}^T \mathbf{B} = \begin{bmatrix} 35 & 10 \\ 13 & 18 \end{bmatrix}$

```
A <- matrix(data = c(1,2,3,5,4,0), nrow = 3, ncol = 2, byrow = TRUE)
```

```
B <- matrix(data = c(4,1,7,4,2,0), nrow = 3, ncol = 2, byrow = FALSE)
```

```
t(A) %*% B
```

```
##      [,1] [,2]
## [1,]   35   10
## [2,]   13   18
```

d. Using the same matrices as in part c, $\mathbf{B}^T \mathbf{A} = \begin{bmatrix} 35 & 13 \\ 10 & 18 \end{bmatrix}$

```
t(B) %*% A
```

```
##      [,1] [,2]
## [1,]   35   13
## [2,]   10   18
```

Show that **A** and **B** are inverses: $\begin{bmatrix} 1 & 2 & 1 \\ 2 & 2 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0.5 & 1 \\ 1 & 0 & -1 \\ 0 & -0.5 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

```
A <- matrix(data = c(1,2,1,2,2,0,1,1,1), nrow = 3, ncol = 3, byrow = TRUE)
```

```
#Use the solve() to find the inverse of a matrix
solve(A)
```

```
##      [,1] [,2] [,3]
## [1,]  -1  0.5   1
## [2,]   1  0.0  -1
## [3,]   0 -0.5   1
```

```
B <- matrix(data = c(-1, 0.5, 1, 1, 0, -1, 0, -0.5, 1), nrow = 3, ncol = 3, byrow = TRUE)
```

```
#Use the solve() to find the inverse of a matrix
solve(B)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    1
## [2,]    2    2    0
## [3,]    1    1    1
```

```
A %*% B
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    1
```

3. Suppose that A is a 4×3 matrix and B is a 3×8 matrix.

```
A <- matrix (data= (1:12), nrow = 4, ncol = 3, byrow = FALSE)
```

```
B <- matrix (data = (1:24), nrow = 3, ncol = 8, byrow = TRUE)
```

a. **AB** exists and is 4×8 matrix.

```
A %*% B
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]  199  214  229  244  259  274  289  304
## [2,]  226  244  262  280  298  316  334  352
## [3,]  253  274  295  316  337  358  379  400
## [4,]  280  304  328  352  376  400  424  448
```

```
dim(A%*%B)
```

```
## [1]  4  8
```

b. **BA** does not exist.

```
# The following produce errors
```

```
# B %*% A
```

4. The determinant of $\det \begin{bmatrix} 1 & -2 \\ 4 & 3 \end{bmatrix} = 11$

```
M5 <- matrix(data = c(1, -2, 4, 3), nrow = 2, ncol = 2, byrow = TRUE)
```

```
det(M5)
```

```
## [1] 11
```

Variables : Types and Summaries

Summary statistics for central tendency

```
dat <- c(1, 2, 3, 5, 5, 4, 6, 6, 1, 2, 10, 11, 9, 9, 9)
```

```
# Mean  
mean(dat)
```

```
## [1] 5.533333
```

```
# Median  
median(dat)
```

```
## [1] 5
```

```
# Mode -- R does not have a built in function for mode but you can use the  
# table function to see what the most common value is  
table(dat)
```

```
## dat  
##  1  2  3  4  5  6  9 10 11  
##  2  2  1  1  2  2  3  1  1
```

Summary statistics for spread

```
# Range  
range(dat)[2]-range(dat)[1]
```

```
## [1] 10
```

```
diff(range(dat))
```

```
## [1] 10
```

```
# Sample Variance  
var(dat)
```

```
## [1] 11.55238
```

```
# Standard Deviation  
sd(dat)
```

```
## [1] 3.398879
```

```
sqrt(var(dat))
```

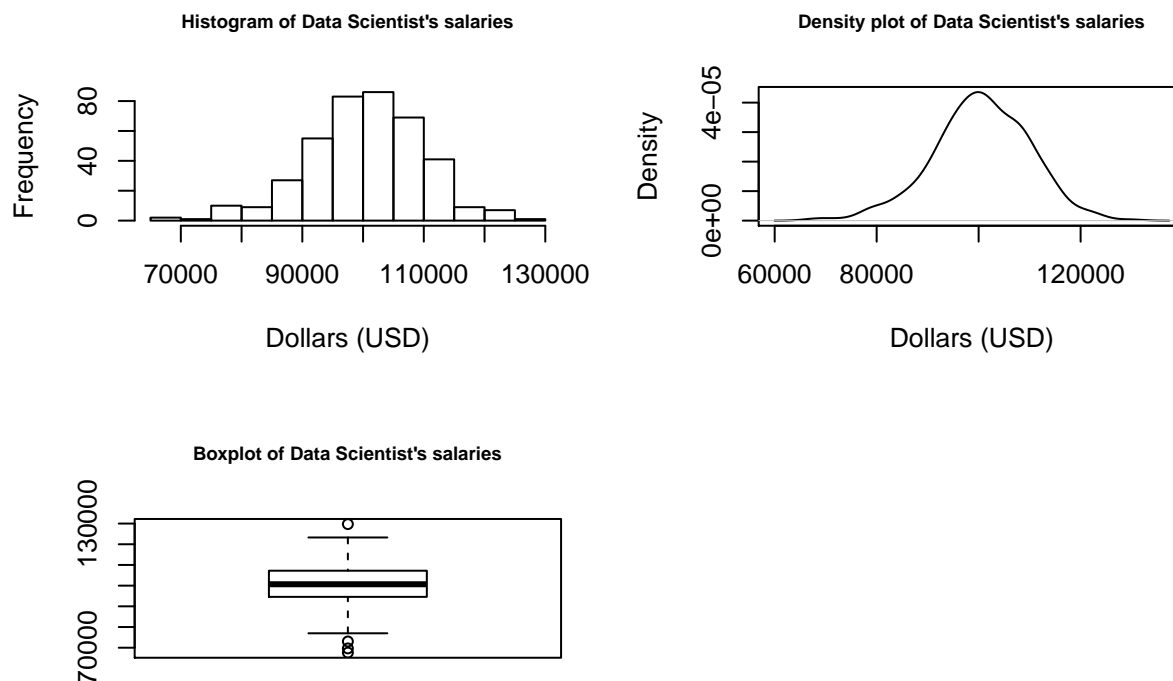
```
## [1] 3.398879
```

Central Limit Theorem Introduction

We can sample from many distributions in R. The following will walk through sampling from a normal distribution.

In the example with average salary of “Data Scientists” we have a sample ($n = 400$), mean of 100,000 and standard deviation of 10,000.

```
DS_salary <- rnorm(n = 400, mean = 1e+05, sd = 10000)
par(mfrow = c(2, 2))
hist(DS_salary, main = "Histogram of Data Scientist's salaries", cex.main = 0.7,
     xlab = "Dollars (USD)", breaks = 10)
plot(density(DS_salary), main = "Density plot of Data Scientist's salaries",
     cex.main = 0.7, xlab = "Dollars (USD)")
boxplot(DS_salary, main = "Boxplot of Data Scientist's salaries", cex.main = 0.7)
```



Confidence Intervals (an example)

```
mean_ <- 100000
sd_ <- 10000
n <- 400
error <- qnorm(0.975)*sd_/sqrt(n)
mean_ - error # lower bound of confidence interval
```

```
## [1] 99020.02
```

```
mean_ + error # upper bound of confidence interval
```

```
## [1] 100980
```

Z-Tests, T-Test, and P-Values

$$H_0 = \mu_{\text{school}} = \mu_{\text{population}} \quad H_a = \mu_{\text{school}} \neq \mu_{\text{population}}$$

Z-Tests and P-Values

‘dnorm’ : probability density function of a normal distribution
‘qnorm’ : computes the probability from a normal distribution
‘pnorm’ : computes the percentile from a normal distribution
‘rnorm’ : randomly draws from a normal distribution

```
# Find the proportion of sample means that is 2 points greater than or less  
# than the state average  
1 - pnorm(72, mean = 70, sd = 1)
```

```
## [1] 0.02275013
```

```
pnorm(72, mean = 70, sd = 1, lower.tail = FALSE)
```

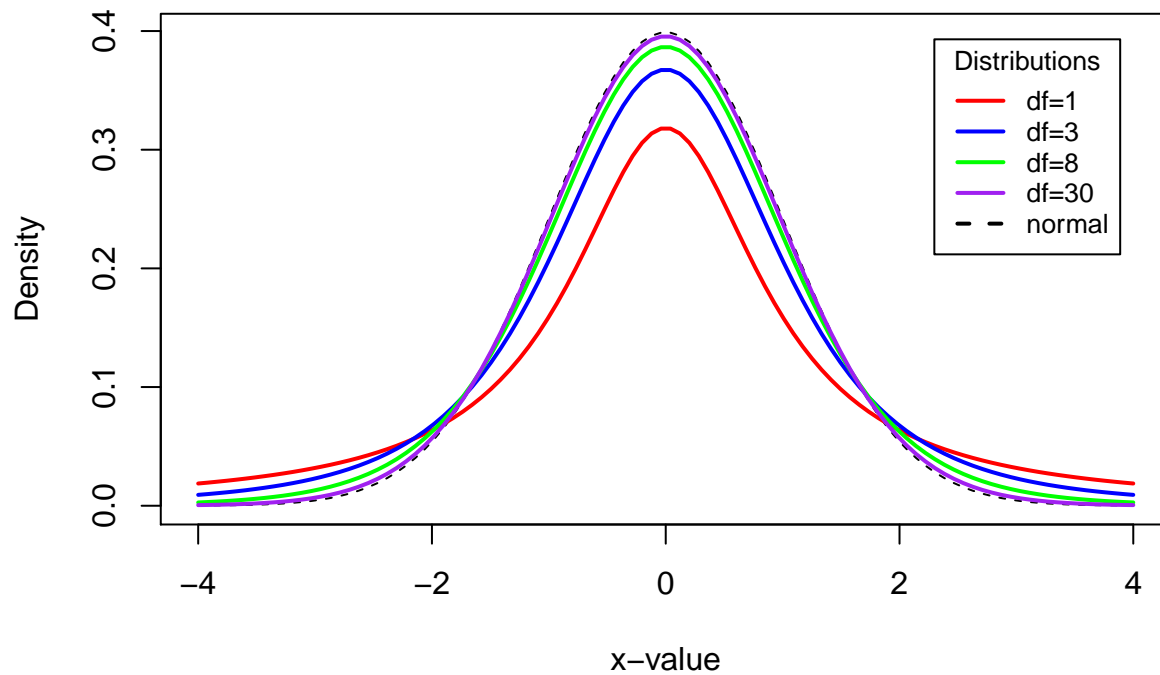
```
## [1] 0.02275013
```

The t-distribution and T-tests

Note: the t distribution is a probability distribution with a single parameter, degrees of freedom (df). It is similar to a standard normal distribution, but has heavier tails when $df < 30$. When $df > 30$, the t distribution is essentially identical to a standard normal distribution. In the example below we plot t-distributions with difference degrees of freedom ($df = 1, 3, 8, 30$) and compare it to normal distribution.

```
x <- seq(-4, 4, length=100)  
zdist <- dnorm(x)  
  
degf <- c(1, 3, 8, 30)  
colors <- c("red", "blue", "green", "purple", "black")  
labels <- c("df=1", "df=3", "df=8", "df=30", "normal")  
  
plot(x, zdist, type="l", lty=2, xlab="x-value",  
      ylab="Density", main="Comparison of t Distributions", cex.main=0.9)  
  
for (i in 1:4){  
  lines(x, dt(x,degf[i]), lwd=2, col=colors[i])  
}  
  
legend("topright", inset=.05, title="Distributions",  
      labels, lwd=2, lty=c(1, 1, 1, 1, 2), col=colors, cex= 0.8)
```

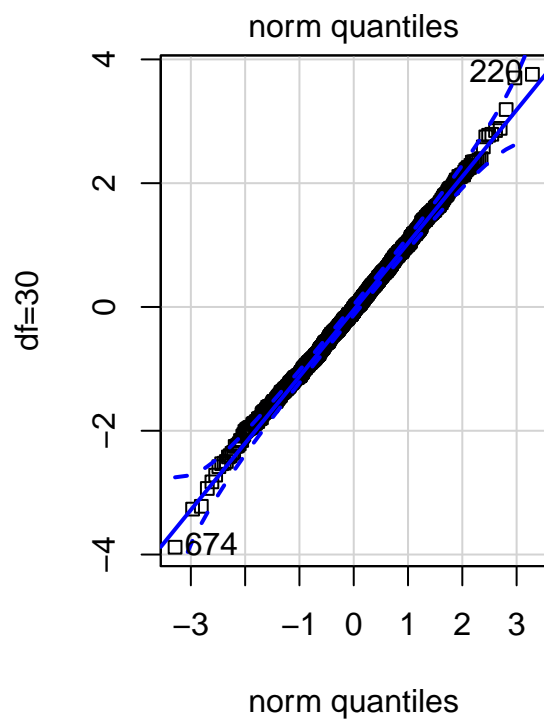
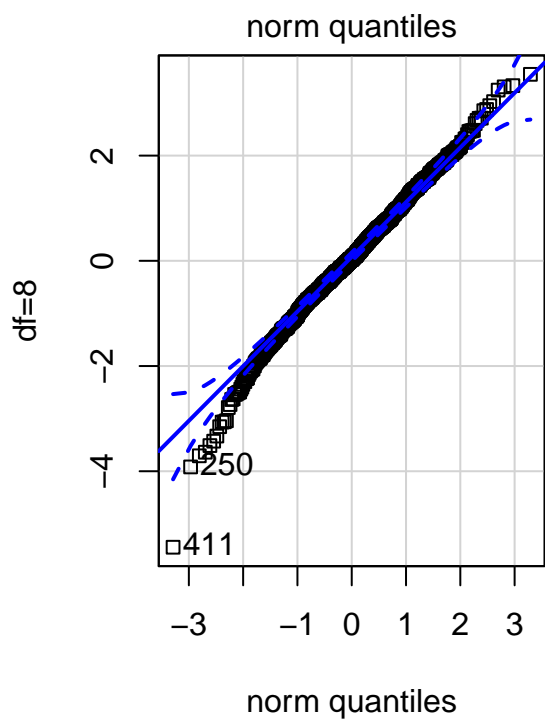
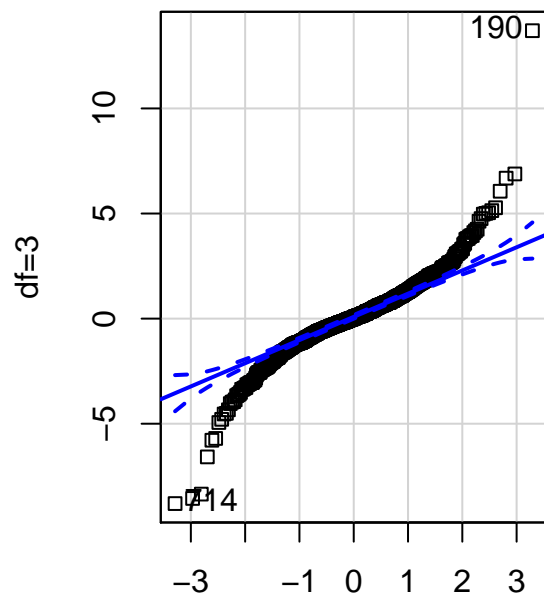
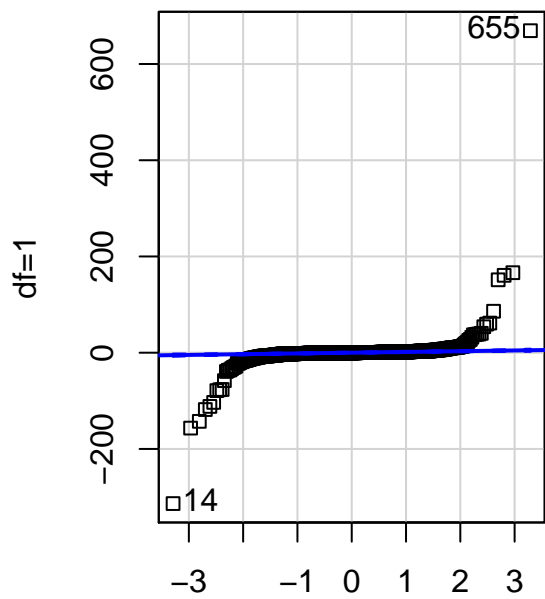
Comparison of t Distributions



We can also look at QQ Plots of data generated from a t-distribution with varying degrees of freedom (using 'rt'). Using the car package and the 'qqPlot' function we see that as the degrees of the freedom increases the tails of the data stay within the the blue lines, indicating it follows a normal distribution.

```
par(mfrow=c(1,2))

for (i in 1:4){
  car::qqPlot(rt(1000, df = degf[i]), ylab = labels[i], pch=22)
}
```



Correlation and Covariance

We can use the ‘cov’ and ‘cor’ functions to find the covariance and correlation of two variables, respectively.

```
set.seed(102)
x <- rnorm(100)
y <- rnorm(100)
z <- x+rnorm(100,0,.4)
w <- -x+rnorm(100,0,.4)
cov(x,z) # Positive covariance
```

```
## [1] 1.162742
```

```
cov(x,y) # No covariance
```

```
## [1] 0.1178042
```

```
cov(x,w) # Negative covariance
```

```
## [1] -1.158946
```

```
cor(x,z) # Positive correlation
```

```
## [1] 0.9508135
```

```
cor(x,y) # No covariance
```

```
## [1] 0.1035828
```

```
cor(x,w) # Negative covariance
```

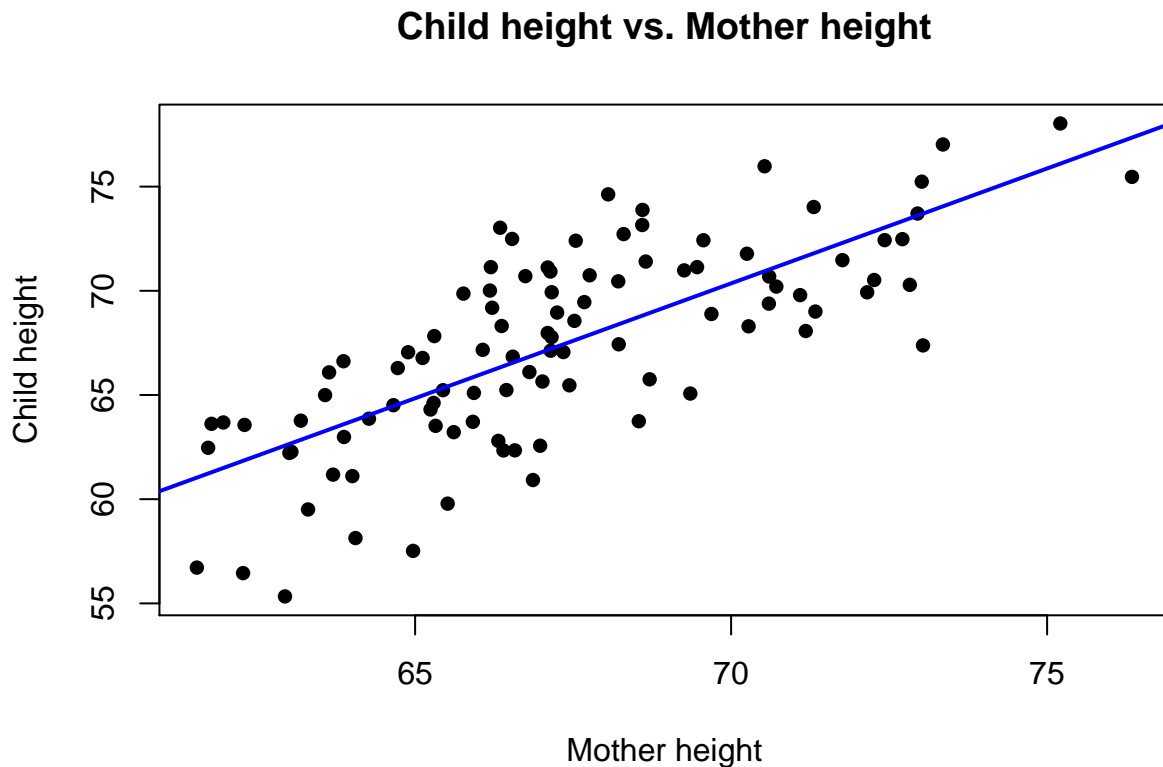
```
## [1] -0.9473492
```

Simple Ordinary Least Squares Regression

```
set.seed(102)
child_height <- rnorm(100, 67, 3)
mother_height <- child_height + rnorm(100, 0, 3)

# We can use lm() to fit a linear model for example lm(y~x) and store the
# fit
fit1 <- lm(mother_height ~ child_height)

plot(child_height, mother_height, pch = 16, main = "Child height vs. Mother height",
      xlab = "Mother height", ylab = "Child height")
abline(fit1, col = 4, lwd = 2)
```



```
# The summary() function produces the summary results of fitted models
summary(fit1)
```

```
##
## Call:
## lm(formula = mother_height ~ child_height)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -7.288 -2.161 -0.091  2.248  6.700
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -6.83595    6.71822  -1.018   0.311
## child_height   1.10276    0.09967  11.064 <2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
##  
## Residual standard error: 3.186 on 98 degrees of freedom  
## Multiple R-squared:  0.5554, Adjusted R-squared:  0.5508  
## F-statistic: 122.4 on 1 and 98 DF,  p-value: < 2.2e-16
```