```r
library(tidyverse)
library(MASS)

#############
#
# Task 9
#
#############


# (a)
m=30
n=12
p=0.7
x<-rbinom(n=m,size=n,prob=p)

# (b)
freq.table=table(x)
# freq.table is of class "table"
observed.vals=as.numeric(names(freq.table))

# add all possible values (not just observed values) to the vector
plot.vals=0:max(observed.vals)

comb.table=matrix(0,2,length(plot.vals))
colnames(comb.table)=plot.vals
rownames(comb.table)=c("sample",sprintf("Binom(%2.1i, %2.1f) pmf",n, p))
# compute proportions for the observed values
comb.table[1,observed.vals+1]=freq.table/m
# compute probabilities for all possible values
comb.table[2,]=dbinom(plot.vals,size=n,prob=p)

barplot(comb.table,beside=TRUE,col=c("blue","red"),ylab="relative frequency / probability",xlab="value",legend.text=TRUE,args.legend=list(x="topleft"))

# (c)
x.bar=mean(x)
s2=var(x)
title(main = sprintf("mean=%2.1f - s^2=%2.1f",x.bar,s2))
T_mnp=sqrt(m)*((x.bar-n*p)/sqrt(n*p*(1-p)))
print(T_mnp)

# (d)
calc.T<-function(m, n, p){
  x<-rbinom(n=m,size=n, prob=p)
  x.bar=mean(x)
  s2=var(x)
  T_mnp=sqrt(m)*((x.bar-n*p)/sqrt(n*p*(1-p)))
  return(T_mnp)
}

# (e)
k=10000
m.vec=c(5,30,500)
size=12
prob = 0.7
z = seq(-4,4,0.01)
for(m in m.vec){
  T.vec=c()
  for(i in 1:k){
    T.vec=c(T.vec,calc.T(m,n,p))
  }
  hist(T.vec,nclass=16,freq=FALSE)
  lines(z,dnorm(z),col="red",lty=3)
}


#############
#
# Task 10
#
#############

# (a)
solar = read.csv2("Solar.csv", stringsAsFactors = TRUE)
solar$batch = as.factor(solar$batch)

# (b)
# create a scatterplot matrix using an argument of class formula
pairs(~Pmax+Imax+Umax+Isc+Uoc,data=solar,col=c("red","blue","green","orange")[solar$batch])

# (c)
# create a boxplot using an argument of class formula
boxplot(Uoc~batch, data=solar,xaxt="n",main="Boxplot for Uoc")
axis(1, at=1:4)

# alternative using the package ggplot2
ggplot(solar)+geom_boxplot(aes(Uoc, col=batch))

# (d)
plot(solar$Pmax,solar$Isc,col=c("red","blue","green","orange")[solar$batch])
legend(x="topleft", legend = levels(solar$batch), col=c("red","blue","green","orange"), pch=1)
X = solar$Pmax[!is.na(solar$Isc)]
Y = solar$Isc[!is.na(solar$Isc)]

# (i)
# see Example I.4.6 and Theorem I.4.9
B = cbind(rep(1, length(X)), X) # alternatively cbind(1, X)
reg.par.mat = ginv(B)%*%Y

# (ii)
reg = lm(Y~X)
reg.par.lm = reg$coef

# comparison of the fitted parameters
reg.par.mat
reg.par.lm

abline(reg.par.lm,col="black")

# (e)
# batch 1
X.1 = solar$Pmax[!is.na(solar$Isc)&solar$batch=="1"]
Y.1 = solar$Isc[!is.na(solar$Isc)&solar$batch=="1"]
reg1 = lm(Y.1~X.1)
reg.par1= reg1$coefficients
abline(reg.par1,col="red")

# batch 4
X.4 = solar$Pmax[!is.na(solar$Isc)&solar$batch=="4"]
Y.4 = solar$Isc[!is.na(solar$Isc)&solar$batch=="4"]
reg4 = lm(Y.4~X.4)
reg.par4= reg4$coefficients
abline(reg.par4,col="orange")

# alternative using the package ggplot2
ggplot(solar)+geom_point(aes(Pmax,Isc,col=batch))+geom_abline(aes(intercept=reg.par.lm[1],slope=reg.par.lm[2]))+geom_abline(aes(intercept=reg.par1[1],slope=reg.par1[2]),col="red")+geom_abline(aes(intercept=reg.par4[1],slope=reg
.par4[2]),col="orange")+scale_color_manual(values=c("red", "blue", "green", "orange"))

# (f)
solar$Isc = round(ifelse(is.na(solar$Isc),predict(reg,newdata=data.frame(X=solar$Pmax)),solar$Isc), digits=3)

# alternative using the package dplyr
solar %>% mutate(
  Isc=round(ifelse(is.na(Isc),predict(reg,newdata=data.frame(X=solar$Pmax)),Isc),digits=3)
)

# (g)
save(solar,file="Solar.RData")



#############
#
# Task 11
#
#############

# (a)
rent.data = read.csv2("rent.csv", stringsAsFactors = TRUE)

# (b)
plot(rent.data$space,rent.data$rent.sqm)
rent.lm1 = lm(rent.sqm~space,data=rent.data)
abline(rent.lm1,col="red")
# the simple linear regression obviously does not capture the structure behind the data
# => transform the data points to find a better model

# the scatterplot is similar to a plot of f(x)=1/x
# => use the reciprocal of space instead:
plot(1/rent.data$space,rent.data$rent.sqm)
# we can see a clear linear structure and model it now using a linear regression

# (c)
rent.lm2 = lm(rent.sqm~I(1/space),data=rent.data)
# the operator / is used in "formula" to construct the design matrix of complex models
# => the formula rent.sqm~1/space is identical to rent.sqm~1
# Use the function I if you want to use an expression of a variable, e.g. 1/space, as predictor
# here this leads to the predictor I(1/space)
abline(rent.lm2,col="red")
# the linear model with the predictor 1/space

plot(rent.data$space,rent.data$rent.sqm)
abline(rent.lm1,col="red")
curve(rent.lm2$coefficients[1]+rent.lm2$coefficients[2]/x,col="blue",add=TRUE)
# the new regression has a clearly better fit

#############
#
# Task 12
#
#############

# (a)
cars.data=read.table(file="cars2.dat",header=TRUE,stringsAsFactors = TRUE)

speed=cars.data$speed
dist=cars.data$dist

# (b)
plot(speed,dist)

# (c)
cars.lm1 = lm(dist ~ speed+I(speed^2))
# Similar to task 11:
# the operators ^ and ** are used in "formula" to construct the design matrix of complex models
# => the formula dist ~ speed+speed^2 is identical to dist ~ speed
# Use the function I if you want to use an expression of a variable, e.g. speed^2, as predictor
# here this leads to the predictor speed+I(speed^2)

par = cars.lm1$coefficients

# curve interprets the first argument as function of "x" and can evaluate it
curve(par[1]+par[2]*x+par[3]*x*x,col="red",add=TRUE)

# alternative solution

# better use the following way - it is numerically more stable because orthogonal polynomials are used
cars.lm2 = lm(dist ~ poly(speed,degree = 2))

# Notice that the parameters are quite different:
cars.lm1$coefficients
cars.lm2$coefficients

# Hence, we cannot simply plug the coefficients of cars.lm2 in a polynomial of degree 2

# Use the function predict:
speed.grid = seq(min(speed),max(speed),length.out = 100)
y.pred = data.frame(speed=speed.grid) # artificial data set with the same predictor variable as cars.lm2
y = predict(cars.lm2,newdata=y.pred)
plot(speed,dist)
lines(speed.grid,y,col="blue")

# (d)
alpha = 0.05
Y = dist
r = 3
r0 = 2
n = 50
B0 = cbind(1,speed)
B = cbind(B0,speed^2)
Q = B%*%solve(t(B)%*%B,t(B))
Q0 = B0%*%solve(t(B0)%*%B0,t(B0))

numerator = Y%*%(Q-Q0)%*%Y/(r-r0)
denominator = Y%*%(diag(50)-Q)%*%Y/(n-r)
F.statistic = numerator/denominator
reject = F.statistic > qf(1-alpha,1,47)
print(reject)

# Bonus: Alternatives in (d)

# compute the p-value:
(p.value <- 1-pf(F.statistic,1,47))
reject.alt = p.value < alpha
print(reject.alt)

# use R functionality
cars.lm3 = lm(dist ~ poly(speed,degree = 1))
anova(cars.lm3,cars.lm2)

summary(cars.lm1)
summary(cars.lm2)
# Note the difference and pay attention if using the results given in summary
```