

HW3Sol

Problem 1

Use ifelse statement to rename state in the murders dataset.

```
library("dslabs")
data(murders)
head(murders)
```

```
##      state abb region population total
## 1  Alabama AL  South   4779736    135
## 2  Alaska  AK   West    710231     19
## 3  Arizona AZ   West   6392017    232
## 4  Arkansas AR  South   2915918     93
## 5 California CA  West  37253956   1257
## 6  Colorado CO   West   5029196     65
```

```
new_names<-ifelse(nchar(murders$stat)>8, murders$abb, murders$state)
new_names
```

```
## [1] "Alabama" "Alaska" "Arizona" "Arkansas" "CA" "Colorado"
## [7] "CT" "Delaware" "DC" "Florida" "Georgia" "Hawaii"
## [13] "Idaho" "Illinois" "Indiana" "Iowa" "Kansas" "Kentucky"
## [19] "LA" "Maine" "Maryland" "MA" "Michigan" "MN"
## [25] "MS" "Missouri" "Montana" "Nebraska" "Nevada" "NH"
## [31] "NJ" "NM" "New York" "NC" "ND" "Ohio"
## [37] "Oklahoma" "Oregon" "PA" "RI" "SC" "SD"
## [43] "TN" "Texas" "Utah" "Vermont" "Virginia" "WA"
## [49] "WV" "WI" "Wyoming"
```

Problem 2

Write a function named compute_s_n.

```
# Write a function compute_s_n with argument n that for any given n computes the sum of 1
↪ + 2^2 + ... + n^2
compute_s_n <- function(n){
  x <- 1:n
  sum(x^2)
}
# Report the value of the sum when n=10
compute_s_n(10)
```

```
## [1] 385
```

Problem 3

(a) and (b)

```
##simulate 30 samples from N(0,1)
set.seed(313)
sam1=rnorm(30)
##equally split the sample, one as the treatment group and the other
##as the control group
group<-rep(c("grp1", "grp2"), each=15)
#### Shuffle the group label
sgroup<-sample(group)
group1=sam1[which(sgroup=="grp1")]
group2=sam1[which(sgroup=="grp2")]
```

- (c) The p-value is 0.169, indicating that there is no sufficient evidence to conclude that the means of the two groups are different.

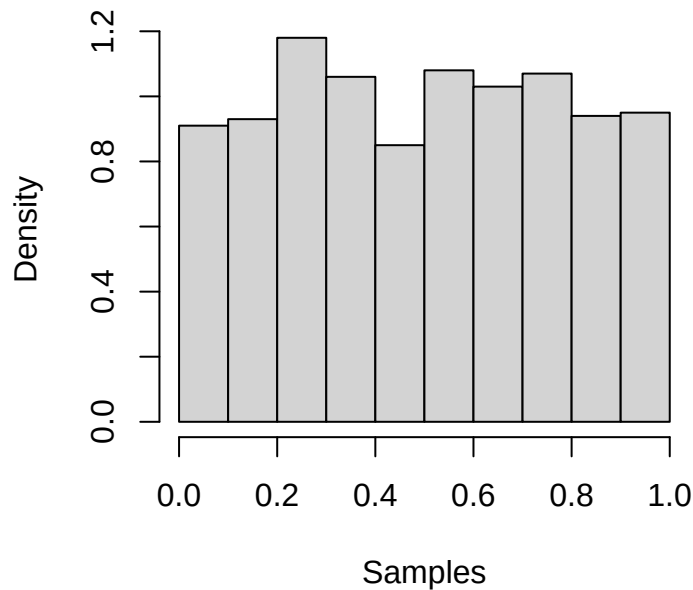
```
##perform t.test for the two samples
t.test(group1,group2)
```

```
##
## Welch Two Sample t-test
##
## data: group1 and group2
## t = -1.4218, df = 22.323, p-value = 0.1689
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
## -1.0674965 0.1987106
## sample estimates:
## mean of x mean of y
## -0.3308338 0.1035591
```

- (d) Histogram for simulation of uniform distribution

```
##generate 1000 samples from uniform distribution
hist(runif(1000),freq=F,main="Histogram of 1000 random samples from uniform",
     ↪ xlab="Samples")
```

histogram of 1000 random samples from un



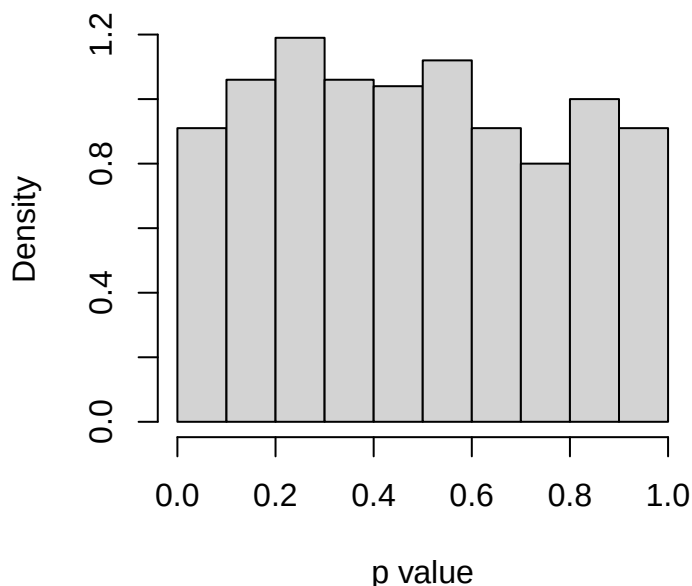
- (e) Under the null hypothesis $H_0 : \mu_1 = \mu_2$ where μ_i is the mean of group i , p values are uniformly distributed. It's equally likely to get any p value between 0, and 1.

```
##p-value of 1000 t-tests
getpvalue=function(iteration){
  g1=sample(1:30,15);g2=(1:30)[-g1]
  group1=sam1[g1]; group2=sam1[g2]
  ##perform t.test for the two samples
  return(t.test(group1,group2)$p.value)
}
```

```
simpvalue<-sapply(1:1000, getpvalue)
```

```
##histogram
hist(simpvalue,xlim=c(0,1),freq = F, main=" Distribution of p values under the
↪ null",xlab="p value")
```

Distribution of p values under the null



The distribution of p value is uniform under the null. Note here that the p-value is a random variable, call it Z . The p-value is computed by calculating the probability of seeing a t-statistic or something more extreme under the null hypothesis. The t-statistic comes from a random variable T that is a transformation of the random variable $\bar{X}$: $T = (\bar{X} - \mu)/(s/\sqrt{n})$. This random variable T has a CDF F .

So, if we can prove the above proposition, we have shown that the p-value's distribution under the null hypothesis is $Uniform(0, 1)$. Let $Z = F(T)$.

$$P(Z \leq z) = P(F(T) \leq z) = P(F^{-1}F(T) \leq F^{-1}(z)) = P(T \leq F^{-1}(z)) = F(F^{-1}(z)) = z.$$

Since $P(Z \leq z) = z$, Z is uniformly distributed, that is, $Uniform(0, 1)$.

Problem 4 Wrting a k-tuple function

```
##(a)
string<-sample(c('A','C','G','T'),10000,replace=TRUE)
##(b)
table(string)
```

```
## string
##      A      C      G      T
## 2433 2519 2520 2528
```

```
##(c)
ktuple<-function(x, k){
  ans<-NULL
  for (i in 1:(length(x)-k+1)){
    tmp<-paste(x[i:(i+k-1)], collapse="")
    ans<-c(ans, tmp)
  }
  tab<-table(ans)
  return(tab)
}
```

```
ktuple(string, k=3)
```

```
## ans
## AAA AAC AAG AAT ACA ACC ACG ACT AGA AGC AGG AGT ATA ATC ATG ATT CAA CAC CAG CAT
## 141 146 143 145 158 158 166 148 149 140 151 162 160 155 164 146 124 183 127 155
## CCA CCC CCG CCT CGA CGC CGG CGT CTA CTC CTG CTT GAA GAC GAG GAT GCA GCC GCG GCT
## 147 170 168 167 166 161 148 168 159 140 158 177 151 154 170 173 142 161 154 156
## GGA GGC GGG GGT GTA GTC GTG GTT TAA TAC TAG TAT TCA TCC TCG TCT TGA TGC TGG TGT
## 166 148 158 150 152 165 165 155 159 148 162 152 142 163 155 163 167 164 165 157
## TTA TTC TTG TTT
## 150 163 166 152
```

Problem 5 Writing a factorial function

Version 1

```
myfactorial<-function(x){
  if (x <0)
    stop("x needs to be an integer > 0")
  if (x==0){
    ans<-1
  } else{
    ans<-1
    for (i in 1:x){
      ans<-ans*i
    }
  }
  return(ans)
}

myfactorial(10)
```

```
## [1] 3628800
```

Version 2

```
myfac2<-function(x){
  if (x <0)
    stop("x needs to be an integer > 0")
  if (x==0){
    ans<-1
  } else{
    txt<-paste(1:x, collapse="*")
    print(paste(txt, "=", sep=""))
    ans<-eval(parse(text=txt))
  }
  return(ans)
}

myfac2(10)
```

```
## [1] "1*2*3*4*5*6*7*8*9*10="
```

```
## [1] 3628800
```