Introduction to R and MAPLE

1 Useful commands in R

1.1 Using R as a Calculator

```
> 2 + 2
[1] 4
> (2 + 2)*5
[1] 20
> 2 + 2 * 5
[1] 12
> 2 + 3 * 5^2
[1] 77
> 3^2 + 4^2
[1] 25
> (4/2)^(1/2)
[1] 1.414214
> sqrt(3)
[1] 1.732051
> 1 + 2 + 3 + 4^2
[1] 22
> (1 + 2 + 3 + 4)^2
[1] 100
> a = 1 + 2 + 3 + 4
> a
[1] 10
> a^2
[1] 100
> sqrt(a)
[1] 3.162278
```

1.2 Defining a Vector

```
> y = 25
```

```
> y
[1] 25
> b <- 1 + 2 + 3; b
[1] 6
> b.test = b + 1; b.test
[1] 7
> b.test = 0; b.test
[1] 0
> Y
Error: Object "Y" not found
> d < - 1 + 2
Error: Object "d" not found
> v1 = c(7, 2, 3, 5); v1
[1] 7 2 3 5
> v2 = numeric(4); v2
[1] 0 0 0 0
> numeric(10)
[1] 0 0 0 0 0 0 0 0 0 0
> v3 = rep(3, 4); v3
[1] 3 3 3 3
> rep(4, 3)
[1] 4 4 4
> v4 = 1:4; v4
[1] 1 2 3 4
> seq(1, 2.2, by=.1)
[1] 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0 2.1
[13] 2.2
> seq(1, 2.2, length=11)
[1] 1.00 1.12 1.24 1.36 1.48 1.60 1.72 1.84 1.96 2.08
[11] 2.20
> v5 = c(v3, v4, 7); v5
[1] 3 3 3 3 1 2 3 4 7
```

1.3 Using Vectors

1.3.1 Simple Arithmetic with Vectors

```
> v1; w1 = 3*v1; w1
[1] 7 2 3 5
[1] 21 6 9 15
> w2 = v1/2; w2
[1] 3.5 1.0 1.5 2.5
> w3 = 5 - v1; w3
[1] -2 3 2 0
> w4 = w3^2; w4
[1] 4 9 4 0
> w5 = w3 + w4; w5
[1] 2 12 6 0
> (5:1)*(1:5)
[1] 5 8 9 8 5
> (5:0)^(0:5); (5:1)/(1:5)
[1] 1 4 9 8 1 0
[1] 5.0 2.0 1.0 0.5 0.2
> (1:10)/(1:2)
[1] 1 1 3 2 5 3 7 4 9 5
> (1:10)/(1:5)
[1] 1.000000 1.000000 1.000000 1.000000 1.000000 6.000000
[7] 3.500000 2.666667 2.250000 2.000000
> (1:10)/(1:3)
[1] 1.0 1.0 1.0 4.0 2.5 2.0 7.0 4.0 3.0 10.0
Warning message: longer object length
is not a multiple of shorter object length
in: (1:10)/(1:3)
```

1.3.2 Indexes and Assignments

```
> w1; w1[3]
[1] 21 6 9 15
```

```
[1] 9
> w5; w5[9]
[1] 3 3 3 3 1 2 3 4 7
[1] 7
> v2; v2[1] = 6; v2
[1] 0 0 0 0
[1] 6 0 0 0
> v7 = numeric(10); v7
[1] 0 0 0 0 0 0 0 0 0 0
> v7[1:3] = 4:6; v7
[1] 4 5 6 0 0 0 0 0 0 0
> v7[2:4]
[1] 5 6 0
```

1.3.3 Vectors Functions

```
> w2; max(w2)
[1] 3.5 1.0 1.5 2.5
[1] 3.5
> w3; mean(w3)
[1] -2 3 2 0
[1] 0.75
> v1; sum(v1)
[1] 7 2 3 5
[1] 17
> v4; prod(v4)
[1] 1 2 3 4
[1] 24
> v5; length(v5)
[1] 3 3 3 3 1 2 3 4 7
[1] 9
> (sum(w3^2) - (sum(w3)^2)/length(w3)) / (length(w3) - 1)
[1] 4.916667
> var(w3)
```

```
[1] 4.916667
> sqrt(var(w3)); sd(w3)
[1] 2.217356
[1] 2.217356
> sqrt(c(1, 4, 9, 16, 25))
[1] 1 2 3 4 5
> 1:5
[1] 1 2 3 4 5
> cumsum(1:5)
[1] 1 3 6 10 15
> cumsum(5:1)
[1] 5 9 12 14 15
> v5; cumsum(v5)
[1] 3 3 3 3 1 2 3 4 7
[1] 3 6 9 12 13 15 18 22 29
> unique(v5)
[1] 3 1 2 4 7
> s = c(rep(3,5), rep(4,10)); s
[1] 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4
> length(s); unique(s); length(unique(s))
[1] 15
[1] 3 4
[1] 2
> round(2.5)
[1] 2
> round(3.5)
[1] 4
> round(5/(1:3))
[1] 5 2 2
> round(5/(1:3), 3)
[1] 5.000 2.500 1.667
```

1.3.4 Comparisons of Vectors

```
> 1:5 < 5:1
[1] TRUE TRUE FALSE FALSE FALSE
> 1:5 <= 5:1
[1] TRUE TRUE TRUE FALSE FALSE
> 1:5 == 5:1
[1] FALSE FALSE TRUE FALSE FALSE
> 1:4 > 4:1
[1] FALSE FALSE TRUE TRUE
> 1:5 < 4
[1] TRUE TRUE TRUE FALSE FALSE
> w4; x3 = (w4 == 4); x3
[1] 4 9 4 0
[1] TRUE FALSE TRUE FALSE
> mean(x3)
[1] 0.5
> sum(c(T, T, F, F, T, T))
[1] 4
> mean(c(T, T, F, F, T, T))
[1] 0.6666667
> v5; v5[v5 < 3]
[1] 3 3 3 3 1 2 3 4 7
[1] 1 2
> length(v5[v5 < 3]); sum(v5 < 3)
[1] 2
[1] 2
```

1.4 Exploring Infinite Sequences

Example 1 *The Sum of the First n Positive Integers.*

```
> n = 1:50
> s = n*(n+1)/2
> cumsum(n)
```

```
[1] 1 3 6 10 15 21 28 36 45 55
[11] 66 78 91 105 120 136 153 171 190 210
[21] 231 253 276 300 325 351 378 406 435 465
[31] 496 528 561 595 630 666 703 741 780 820
[41] 861 903 946 990 1035 1081 1128 1176 1225 1275
> s
[1] 1 3 6 10 15 21 28 36 45 55
[11] 66 78 91 105 120 136 153 171 190 210
[21] 231 253 276 300 325 351 378 406 435 465
[31] 496 528 561 595 630 666 703 741 780 820
[41] 861 903 946 990 1035 1081 1128 1176 1225 1275
> mean(s == cumsum(n))
[1] 1
> n = 1:50000
> s = n*(n + 1)/2
> mean(s == cumsum(n))
[1] 1
```

Example 2 *A Sequence with Limit* $e = 2.71828$ *is the limit of the infinite sequence* $a_n = (1 + 1/n)^n$*, where* $n = 1, 2, 3. \ldots$.

```
> n = 1:10000; a = (1 + 1/n)^n
> cbind(n, a)[c(1:5, 10^(1:4)), ]
n a
[1,] 1 2.000000
[2,] 2 2.250000
[3,] 3 2.370370
[4,] 4 2.441406
[5,] 5 2.488320
[6,] 10 2.593742
[7,] 100 2.704814
[8,] 1000 2.716924
[9,] 10000 2.718146
```

The sequence a_n is monotone increasing (that is, each term in the sequence is larger than the one before). This can be illustrated by taking successive differences of the vector a.

```
> da = diff(a)
> da[1:10]
[1] 0.25000000 0.12037037 0.07103588 0.04691375
[5] 0.03330637 0.02487333 0.01928482 0.01539028
[9] 0.01256767 0.01045655
> mean(da > 0)
[1] 1
> exp(1) - a[10000]
[1] 0.0001359016
```

Plot the first 200 values of an against the numbers $n = 1, 2, \ldots, 200$.

```
> plot(1:200, a[1:200], pch=19, main="A Sequence That Approaches e")
> abline(h = exp(1), col="darkgreen", lwd=2, lty="dashed")
```

which we choose to draw twice the normal thickness (parameter **lwd**), dashed (**lty**), and in green (**col**).

Example 3 *A Loop to Find the Mean of a Vector.*

```
w3 = c(-2, 3, 2, 0) # omit if w3 defined earlier in your R session
avg = 0 # when program finishes, this will be the mean
n = length(w3)
for (i in 1:n)
{
avg = avg + w3[i]
}
avg = avg/n; avg
> avg
[1] 0.75
```

Example 4 *A Loop to Print Fibonacci Numbers. Find the first 30 elements of the Fibonacci sequence,* $1, 1, 2, 3, 5, 8, 13, \ldots$.

```
m = 30; fibo = numeric(m); fibo[1:2] = 1
for (i in 3:m)
{
fibo[i] = fibo[i-2] + fibo[i-1]
}
fibo
> fibo
[1] 1 1 2 3 5 8 13 21
[9] 34 55 89 144 233 377 610 987
[17] 1597 2584 4181 6765 10946 17711 28657 46368
[25] 75025 121393 196418 317811 514229 832040
```

The sequence of ratios, obtained by dividing each Fibonacci number by the previous one, rapidly approaches the so-called Golden Ratio 1.618 used in ancient Greek architecture.

Example 5 *Find these ratios, print the* $(m-1)$*st value, and make a figure that uses the first 15 ratios to illustrate the rapid approach to a limit.*

```
golden = fibo[2:m]/fibo[1:(m-1)]
golden[m-1]
plot(1:15, golden[1:15],
main="Fibonacci Ratios Approach Golden Ratio")
abline(h=golden[m-1], col="darkgreen", lwd=2)
> golden[m-1]
[1] 1.618034
```

1.5 Graphical Functions

Example 6 *Plotting a Function:* $f(t) = 6t(1-t)$*, where* $0 \leq t \leq 1$*.*

```
t = seq(0, 1, length=200)
f = 6*(t - t^2)
plot(t, f, type="l", lwd=2, col="blue", ylab="f(t)",
main="Density of BETA(2,2)")
```

Example 7 *Making a Histogram of Data.*

```
iq <- c( 84, 108, 98, 110, 86, 123, 101, 114, 121, 131,
90, 108, 105, 93, 95, 102, 119, 98, 94, 73)
hist(iq, xlab="IQ", col="wheat", label=T,
main=paste("Histogram of IQ Scores of", length(iq), "Students"))
```

We have used the data to generate one part of the main header with the character function **paste** and used the argument **labels=T** to print counts atop the bars. The argument **col="wheat"** fills bars with a designated color.

1.6 Sampling from a Finite Population

Example 8 *Poker Hands. Count the Aces in a 5-card poker hand. It is convenient to associate the numbers 1, 2, 3, 4 with the four Aces.*

```
> h = sample(1:52, 5); h
[1] 36 10 2 39 26
> h < 5; sum(h < 5)
[1] FALSE FALSE TRUE FALSE FALSE
[1] 1
set.seed(1234)
m = 100000
aces = numeric(m) # m-vector of 0s to be modified in loop
for (i in 1:m)
{
h = sample(1:52, 5)
aces[i] = sum(h < 5) # ith element of ‘aces’ is changed
}
cut = (0:5) - .5
hist(aces, breaks=cut, prob=T, col="Wheat",
main="Aces in Poker Hands")
summary(as.factor(aces)) # observed counts
summary(as.factor(aces))/m # simulated probabilities
round(choose(4, 1)*choose(48, 4)/choose(52, 5), 4) # P{1 Ace}
> summary(as.factor(aces)) # observed counts
```

```
0 1 2 3 4
65958 29982 3906 153 1
> summary(as.factor(aces))/m # simulated probabilities
0 1 2 3 4
0.65958 0.29982 0.03906 0.00153 0.00001
> round(choose(4, 1)*choose(48, 4)/choose(52, 5), 4) # P{1 Ace}
[1] 0.2995
```

In the hist function, we use two arguments to modify the default version.

- The argument **prob=T** puts "densities" (that is, relative frequencies or proportions out of 100 000) on the vertical axis of the histogram.
- An argument to specify the parameter **breaks** improves the break points along the horizontal axis to make a nice histogram.

Note the command **as.factor** interprets aces as a vector of categories to be tallied by the summary statement.

Example 9 *Rolling Dice. The code below simulates rolling two dice.*

```
> d = sample(1:6, 2, repl=T); d
[1] 6 6
```

If sampling is to be done with replacement, a third argument **repl=T** is required, otherwise sampling is without replacement.

What is the probability of getting the same number on both dice? You could determine whether this happens on a particular simulated roll of two dice with the code **length(unique(d))**.

```
set.seed(1212)
m = 100000; x = numeric(m)
for (i in 1:m)
{
x[i] = sum(sample(1:6, 2, repl=T))
}
```

```
cut = (1:12) + .5; header="Sums on Two Fair Dice"
hist(x, breaks=cut, prob=T, col="wheat", main=header)
summary(as.factor(x))
> summary(as.factor(x))
2 3 4 5 6 7 8 9 10 11 12
2793 5440 8428 11048 13794 16797 13902 11070 8337 5595 2796
```

In this simulation of rolls of pairs of dice, we can avoid writing a loop, as shown in the program below.

```
set.seed(2008)
m = 100000
red = sample(1:6, m, repl=T); green = sample(1:6, m, repl=T)
x = red + green
summary(as.factor(x))
sim = round(summary(as.factor(x))/m, 3)
exa = round(c(1:6, 5:1)/36, 3); rbind(sim, exa)
hist(x, breaks=(1:12)+.5, prob=T, col="wheat",
main="Sums on Two Fair Dice")
points(2:12, exa, pch=19, col="blue")
> summary(as.factor(x))
2 3 4 5 6 7 8 9 10 11 12
2925 5722 8247 11074 13899 16716 13633 11166 8309 5495 2814
> sim = round(summary(as.factor(x))/m, 3)
> exa = round(c(1:6, 5:1)/36, 3); rbind(sim, exa)
2 3 4 5 6 7 8 9 10 11 12
sim 0.029 0.057 0.082 0.111 0.139 0.167 0.136 0.112 0.083 0.055 0.028
exa 0.028 0.056 0.083 0.111 0.139 0.167 0.139 0.111 0.083 0.056 0.028
```

More examples of commands, see http://cran.r-project.org/doc/contrib/Short-refcard.pdf or Robert and Casella (2009), Chapter 1, Figure 1.1—1.4 and Table 1.1.

1.7 Problems

1. What will R codes be?

```
numeric(100); numeric(10); rep(0, 10); rep(10, 10)
seq(0, 10); seq(0, 10.5, by=1); seq(0, 10, length=11)
0:9.5; -.5:10; 0:10 -.5; -1:9 + .5; seq(-.5, 9.5)
-4:11; 4:-1; 4.5:10; -4:-11.5
(10:22)/10; 10:22/10; 10/2:22; (10/2):22
seq(1, 2.2, by=0.1); seq(by=0.1, to=2.2, from=1)
seq(1, 2.2, length.out=13); seq(1,2.2, len=13)
r = 1:5; s = -2:2; s/r; r/s; s/s
r^0; s^0; s^.5;1000^1000; ?NaN
rep("I will not eat anchovy pizza in class.", 20)
rep(1:4, times=3); rep(1:4, each=3)
```

2. Which statements in the following lines of R code produce output? Predict and use R to verify the output.

```
x = 0:10; f = x*(10-x)
f; f[5:7]
f[6:11] = f[6]; f
x[11:1]
x1 =(1:10)/(1:5); x1; x1[8]
x1[8] = pi; x1[6:8]
```

3. Predict and verify results of the following. The function diff of a vector makes a new vector of successive differences that has one less element than the original vector. The last line approximates e^2 by summing the first 16 terms of the Taylor (Maclaurin) series $e^x = \sum_{n=0}^{\infty} x^n/n!$, where $x = 2$.

```
length(0:5); diff(0:5); length(diff(0:5)); diff((0:5)^2)
x2 = c(1, 2, 7, 6, 5); cumsum(x2); diff(cumsum(x2))
unique(-5:5); unique((-5:5)^2); length(unique((-5:5)^2))
```

```
prod(1:5); factorial(5); factorial(1:5)
exp(1)^2; a1 = exp(2); a1
n = 0:15; a2 = sum(2^n/factorial(n)); a2; a1 - a2
```

4. The functions **floor** (to round down) and **ceiling** (to round up) work similarly to **round**. You should explore these functions, using the vectors shown above to illustrate round. See **?round** for other related functions.

5. Predict and verify the results of the following statements. Which ones produce output and which do not? The last two lines illustrate a grid search to approximate the maximum value of $f(t) = 6(t - t^2)$, for t in the closed interval $[-1, 1]$. Show that if t is chosen to have length 200, instead of 201, then the result is neither exact nor unique.

```
x4 = seq(-1, 1, by = .1); x5 = round(x4); x4; x5
unique(x5); x5==0; x5[x5==0]; sum(x4==x5)
sum(x5==0); length(x5); mean(x5==0); x5 = 0; x5
t = seq(0, 1, len=201); f = 6*(t - t^2)
mx.f = max(f); mx.f; t[f==mx.f]
```

6. With $s_n = n(n+1)/2$, use R to illustrate the limit of s_n/n^2 as $n \to \infty$.

7. Present the numerical and graphical displays to illustrate that the sequence $b_n = (1 + 2/n)^n$ converges to e^2.

8. Make a plot of the density function of the distribution $BETA(3.2)$.

2 Useful Commands in MAPLE

Example 10 *Calculate the sample mean and sample variance of n random numbers*

```
> hw1:=proc(n)
> local i,a,total1,total2,xbar,var;
> a:=array(1..n);
> total1:=0;
> total2:=0;
> for i from 1 to n do
>    a[i]:=stats[random, uniform[0,1]](1);
>    total1:=total1+a[i];
>    total2:=total2+a[i]^2;
> od;
> xbar:=evalf(total1/n);
> var:=evalf((total2-n*xbar^2)/(n-1));
> print(xbar,var);
> end;

> hw2:=proc(n)
> local i,a,total1,total2,xbar,var;
> a:=array(1..n);
> total1:=0;
> total2:=0;
> for i from 1 to n do
>    a[i]:=i;
>    total1:=total1+a[i];
>    total2:=total2+a[i]^2;
> od;
> xbar:=evalf(total1/n);
> var:=evalf((total2-n*xbar^2)/(n-1));
> print(xbar,var);
> end;
> hw2(10);
```

5.500000000, 8.250000000

Example 11 *Calculate the distribution of Poisson*

```
> with(stats):
> stats[statevalf,pf,poisson[1]](0);#P(X=0)
                              0.3678794412
> exp(-1.0);
                              0.3678794412
> stats[statevalf,dcdf,poisson[1.2]](3);#P(X<=3)
                              0.9662310318
> stats[statevalf,dcdf,poisson[1.2]](1);#P(X<2)
                              0.6626272662
> stats[statevalf,dcdf,poisson[1.2]](6)-stats[statevalf,dcdf,poisson[1.2]](3);
#P(3<X<=6)
                              0.0335178557
```

Example 12 *Calculate the mean and variance of discrete uniform in* $[1, 5]$

```
> stats[statevalf,pf,discreteuniform[1,5]](1);
                              0.2000000000
> hw3:=proc(n)
> local i,a,mu,var;
> a:=array(1..10);
> for i from 1 to 10 do
>    a[i]:=i;
> od;
> a:=convert(a,'list');
> mu:=evalf(describe[mean](a));
> var:=evalf(describe[variance](a));
> print(mu,var);
> end;
> hw3(10);
                        5.500000000, 8.250000000
```