

Solutions to Exercise Series 2

1. **Vectors.** No solution given here.

2. **Selecting elements** *From the data.frame d.sport, select*

a) *the values of all athletes in the discipline kugel*

```
> d.sport[, 'kugel']
[1] 15.66 13.60 15.82 15.31 16.32 14.01 13.53 14.71 16.91 15.57
[11] 14.85 15.52 16.97 14.69 14.51
```

b) *the performances of athletes SMITH and MUELLER for the first 3 disciplines, and their total (punkte).*

```
> d.sport[c('SMITH', 'MUELLER'), c(1:3, 7)]
      weit kugel hoch punkte
SMITH  7.47 16.97  195  8271
MUELLER 7.25 14.69  195  8253
```

c) *the data for all athletes who achieved at least 8600 punkte.*

```
> d.sport[d.sport[, 'punkte']>=8600,]
      weit kugel hoch disc stab speer punkte
OBRIEN  7.57 15.66  207 48.78  500 66.90  8824
BUSEMANN 8.07 13.60  204 45.04  480 66.86  8706
DVORAK   7.60 15.82  198 46.28  470 70.16  8664
FRITZ    7.77 15.31  204 49.84  510 65.70  8644
HAMALAINEN 7.48 16.32  198 49.62  500 57.66  8613
```

3. **Matrices.** *Generate the following matrices.*

a) [,1] [,2]

```
[1,]  1 101
[2,]  2 102
[3,]  3 103
[4,]  4 104
```

and

```
      [,1] [,2] [,3] [,4]
[1,]  1   2   3   4
[2,] 101 102 103 104
```

```
> t.m1 <- matrix(c(1:4,101:104), nrow=4)
> t.m2 <- matrix(c(1:4,101:104), nrow=2, byrow=TRUE)
> t.m1 <- cbind(1:4,101:104) ; t.m2 <- rbind(1:4,101:104)
> t.m2 <- t(t.m1)
```

b) [,1] [,2] [,3] [,4] [,1] [,2] [,3] [,4]

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

```
> t.d1 <- 5*diag(4) ; t.d2 <- diag(c(3, 1, -2, 0))
```

- c) *Multiply t.d1 by t.d2 elementwise.*

```
> t.d1 * t.d2
```

Try the same for t.m1 and t.d2.

```
> t.m1 * t.d2
```

does not work (fortunately), the answer is

```
Error in t.m1/t.d2 : non-conformable arrays
```

But `c(t.m1) * t.d2` would give a result. `c(t.m1)` uses the elements of `t.m1` (columnwise) to generate a simple vector, and vectors are used liberally to make the required operations possible. The elements are recycled to obtain the required length, which for this operation is the number of elements in `t.d2`, which is 16.

- d) *Multiply the two matrices t.m2 and t.d2 by matrix multiplication. Do the same for t.m1 and t.d2.*

```
> t.m2 %% t.d2
```

```
> t.m1 %% t.d2
```

leads to the same error as above,

```
Error in t.m1 %% t.d2 : non-conformable arrays
```

- e) *Matrix-multiply t.d2 by the vector 5:8, from the left as well as from the right.*

```
> 5:8 %% t.d2 ; t.d2 %% 5:8
```

Both operations work. Vectors are used as row or column vectors, whatever leads to a sensible operation.

4. Tables. Use the function `table` to

- a) *count the number of athletes that achieve less than and more than 8600 punkte (according to d.sport).*

```
> t.l <- d.sport[, 'punkte'] > 8600 ; table(t.l)
```

```
t.l
```

```
FALSE TRUE
```

```
10    5
```

... or type directly `table(d.sport[, 'punkte'] > 8600)`

- b) *cross tabulate this with the distinction between those that achieve less or more than 7.8 in weit.*

```
> table(d.sport[, 'punkte'] > 8600, d.sport[, 'weit'] > 7.8)
```

```
FALSE TRUE
```

```
FALSE    8    2
```

```
TRUE     4    1
```

- c) *Use hist with the argument plot=FALSE to obtain a classification of punkte.*

```
> hist(d.sport[, 'punkte'], plot=FALSE)
```

The “components” `$breaks` and counts are

```
$breaks
```

```
[1] 8200 8300 8400 8500 8600 8700 8800 8900
```

```
$counts
```

```
[1] 6 2 1 1 3 1 1
```

That is: 6 athletes obtained between 8200 and 8300 punkte, 2 fall in the next bin (between 8300 and 8400) and so on.

- d) *Read ?cut and try to obtain the same result with cut and table (up to the names of the classes).*

```
> table(cut(d.sport[, 'punkte'], seq(8200,8900,by=100)))
```

```
(8.2e+03,8.3e+03] (8.3e+03,8.4e+03] (8.4e+03,8.5e+03] (8.5e+03,8.6e+03]
```

```
6                2                1                1
```

```
(8.6e+03,8.7e+03] (8.7e+03,8.8e+03] (8.8e+03,8.9e+03]
```

```
3                1                1
```

5. Estimation, Testing, Confidence Interval.

- a) Obtain the mean and the median of the *extra* hours of sleep for group 2.

```
> mean(sleep[sleep[,2]==2,1])
[1] 2.33
> median(sleep[sleep[,2]==2,1])
[1] 1.75
```

- b) Test the hypothesis that the 2 groups sleep essentially equally long by a Wilcoxon rank sum test (*wilcox.test*) and by a *t.test*.

```
> wilcox.test(sleep[sleep[,2]==1,1], sleep[sleep[,2]==2,1], paired=FALSE)
```

```
wilcox.test(sleep[sleep[,2]==1,1], sleep[sleep[,2]==2,1], paired=FALSE)
```

Wilcoxon rank sum test with continuity correction

```
data: sleep[sleep[, 2] == 1, 1] and sleep[sleep[, 2] == 2, 1]
W = 25.5, p-value = 0.06933
alternative hypothesis: true location shift is not equal to 0
```

Warning message:

```
In wilcox.test.default(sleep[sleep[, 2] == 1, 1], sleep[sleep[, 2] == 2, 1]) :
cannot compute exact p-value with ties
```

The same result is obtained from `wilcox.test(extra group, data=sleep)`.

```
> t.test(extra group, data=sleep)
```

Welch Two Sample t-test

```
data: extra by group
t = -1.8608, df = 17.776, p-value = 0.0794
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.3654832  0.2054832
sample estimates:
mean in group 1 mean in group 2
      0.75      2.33
```

- c) Get the confidence interval from the output of *t.test*. What is the parameter for which this confidence interval applies?

$[-3.3654832, 0.2054832]$ is the confidence interval for the difference of *extra* hours between the groups. It contains 0, which is necessary since the p value is > 0.05 .