

Solutions to Exercise Series 5

1. Simulation of Confidence Intervals

- a) The 500 samples of size 10 can be generated with the following command

```
t.stichprob <- matrix(rnorm(500*10,0,2),ncol=10)
```

- b) The 95%-confidence interval for μ based on the t-test is given by

$$\bar{X} \pm t_{0.975,n-1} \cdot \hat{\sigma} / \sqrt{n},$$

where $\hat{\sigma}$ is the empirical standard deviation of the sample X_1, \dots, X_n and $t_{0.975,n-1}$ the 97.5%-quantile of the t-distribution with $n - 1$ degrees of freedom. For $n = 10$ is $t_{0.975,9} = 2.262$ (R-command `qt(0.975,df=9)`).

```
t.a <- t.test(t.stichprob[1,]); str(t.a)
```

The variable name for the confidence interval is `conf.int`.

Define function (as on sheet or like this):

```
f.conf <- function(x)t.test(x)$conf.int
```

Apply function to every line:

```
t.conf <- apply(t.stichprob,1,f.conf)
```

Note: The result for the first line of `t.stichprob` is contained in the first column of `t.conf`.

Calculate how often $\mu = 0$ lies within the confidence interval:

```
sum((t.conf[1,]<0)&(t.conf[2,]>0)) # one way to do this
```

```
sum(t.conf[1,]*t.conf[2,]<0) # same result
```

From theory one would expect that 95% of all 95%-confidence intervals contain the true parameter. The confidence intervals simulated in this solution contained the value " $\mu = 0$ " 471 times, i.e. in ca. 94% of all cases.

- c) Define the function

```
f.konfidenz <- function(n,clev){
  t.stichprob <- matrix(rnorm(n*10,0,2),ncol=10)
  f.conf <- function(x){
    t.test(x,mu=0,alternative='two.sided',
           conf.level=clev)$conf.int
  }
  t.conf <- apply(t.stichprob,1,f.conf)
  sum((t.conf[1,]<0)&(t.conf[2,]>0))/n
}
```

2. Functions for Graphical Output

a) Read in the data

```
d.pcb <-
```

```
read.table("http://stat.ethz.ch/~stahel/courses/R/pcb.txt",header=TRUE)
```

Create a general function that reads in a data frame and automatically determines the size of the plot. Again, there are several ways to do this, here are two examples.

(i) Using the for-loop (usually not recommended, because slower than `apply`). In this example, the size/outline of the plot `par()` is determined within the function definition.

```
f.plot.hist <- function(d.frame){
  vars <- names(d.frame)          # find variable names
  nvars <- length(vars)           # get number of variables,
                                  # also length(d.frame) is possible
  par(mfrow=c(ceiling(nvars/2),2)) # subdivide plot, "ceiling" rounds to
                                  # next higher number
  for(i in 1:length(vars)){       # the for-loop
    hist(x=d.frame[,vars[i]],     # plot histograms
          main=paste("Histogram of ",vars[i]), # for all variables
          xlab=vars[i],ylab="Frequency")
  }
}
```

(ii) It is possible to use `apply` in the same function.

```
f.plot.hist <- function(d.frame){
  vars <- names(d.frame)          # find variable names
  nvars <- length(vars)           # get number of variables,
                                  # also length(d.frame) is possible
  par(mfrow=c(ceiling(nvars/2),2)) # subdivide plot, "ceiling" rounds to
                                  # next higher number

  apply(d.frame,2,hist,
        main="Histogram",ylab="Frequency") # plot histograms
                                             # for all variables
}
```

However, we are not able to label the plots and axes corresponding to the variables when we use `apply` (at least I haven't been able to think of a way). So this is another example where you need to use a `for`-loop to achieve the desired outcome.

Now call the new function using only the non-factor variables of `d.pcb[,2:6]`.

```
f.plot.hist(d.pcb[2:6])
```

(*) This is shown for the for-loop version (i) above. Use if-else statements and is.factor and table to create barplots of the number of occurrences for each factor level. See Fig. 1

```
f.plot.hist <- function(d.frame){
  dfnam <- deparse(substitute(d.frame)) # creates the name of the given
                                         # data frame as character string

  vars <- names(d.frame)
  nvars <- length(vars)
  par(mfrow=c(ceiling(nvars/2),2),oma=c(0,0,3,0)) # a nice way to add a
                                                    # title to the whole page

  for(i in 1:length(vars)){
    if(is.factor(d.frame[,vars[i]])){           # include additional if
      VV <- table(d.frame[,vars[i]])           # generate counts for
                                              # each class
      barplot(VV,main=paste("Barplot of ",vars[i])) # barplot
    } else {
      hist(x=d.frame[,vars[i]],
           main=paste("Histogram of ",vars[i]),
           xlab=vars[i],ylab="Frequency")
    }
  }
  mtext(text=paste("Data Frame ",dfnam),outer=TRUE) # add the page title
}
```

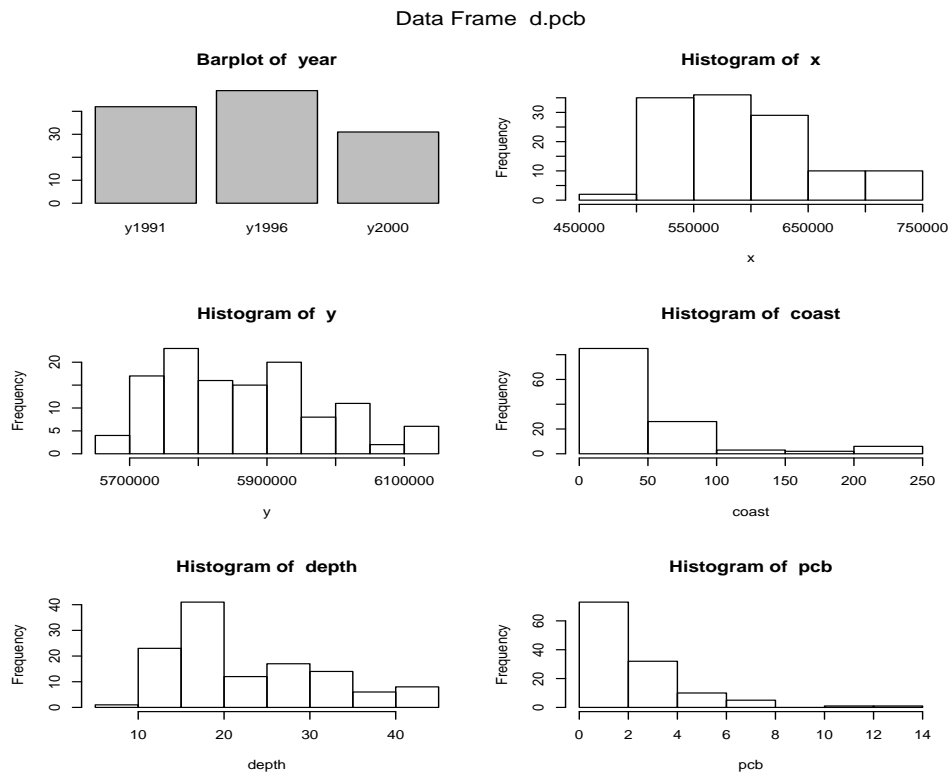


Figure 1: Histograms and Bar plots for all variables of data frame d.pcb

b) Read in new data frame

```
d.sport <-
```

```
  read.table("http://stat.ethz.ch/Teaching/Datasets/NDK/sport.dat",
            header=TRUE)
```

Try your new function `f.plot.hist` on new data (see Fig. 2).

```
f.plot.hist(d.sport)
```

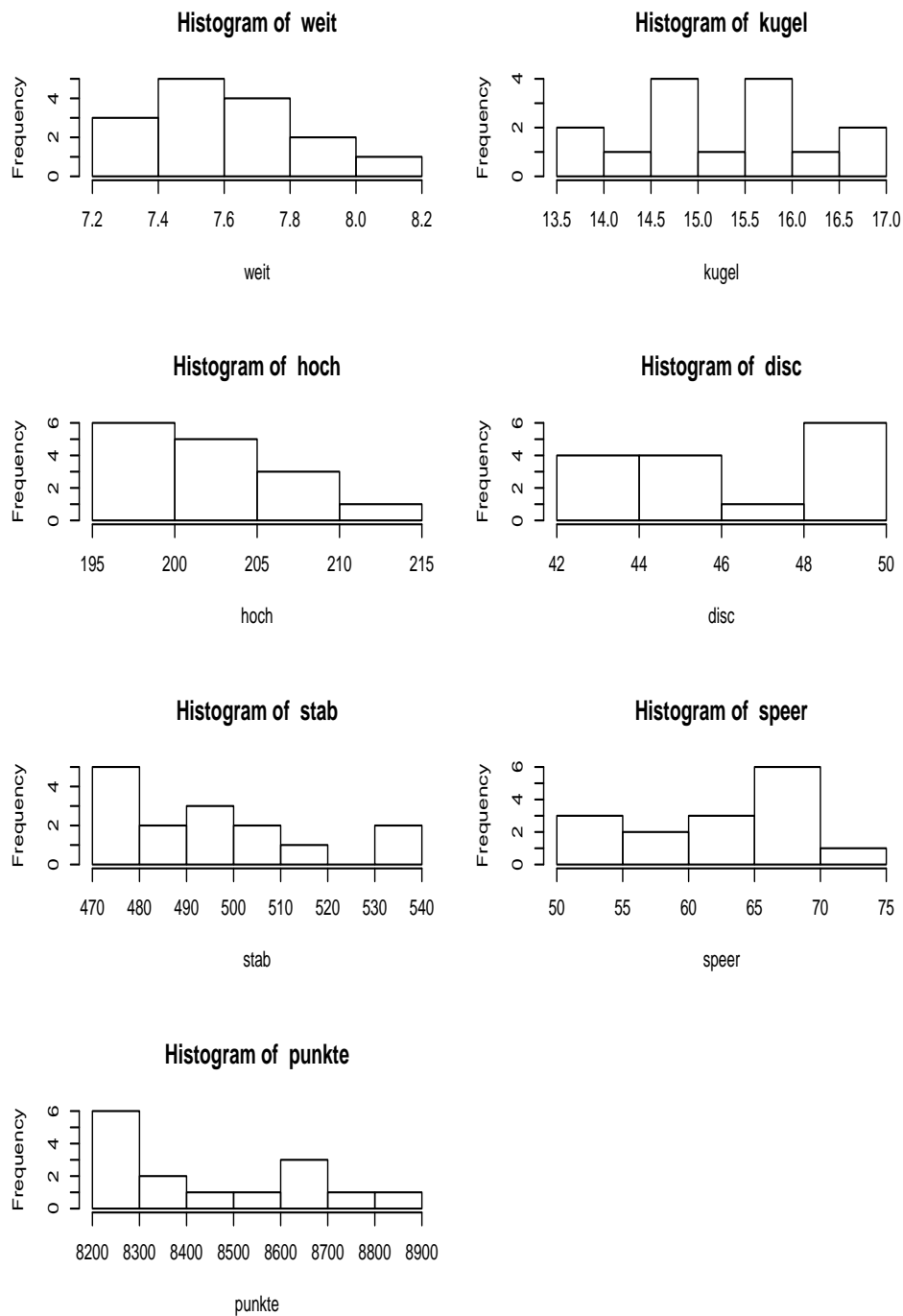


Figure 2: Histogram of data frame `d.sport`