Part 1

Data frames and basic plotting

We will start today’s lab by looking at data collected to study the concentration of some metals in the blood of birds, specifically copper and lead. This will give you a chance to recall and apply some basic functions in R covered by the first two lectures of the course.

1. Download the file “Metals.txt” from the course home page, and read it into a data frame object in R which you call Metals. In the help file for read.table, there are 5 different functions mentioned for reading in a table. Find the one that works correctly on this data, and decide if you should use the option header=TRUE or header=FALSE.

2. Sort the rows of the object Metals so that it becomes alphabetical in the text given in the “Individual” column. The function order is useful here. Also, recall the technique of using the notation Metals[, , ] to select rows of a data frame.

3. Take a few minutes and explore the data for the lead and copper measurements. Use the appropriate R functions to compute simple statistics such as the minimum, maximum, mean, median, variance, and standard deviation. Visualize the data with a simple plot (for example, using the function hist).

4. It seems that there are many small and a few large copper measurements. In such situations, logarithmic transformations are often useful. Transform the data using logarithms with base 2 (see help for the function log), and append this as an extra column in your Metals object. Make sure the new column has an appropriate name. For appending columns, the function cbind is useful.

5. Plot the lead measurements against the logarithm of the copper measurements for the same individual. Is there a strong connection? Give the plot a nice heading, and nice labels for the x and y axes (see help for plot). Also, redo the plot with a different plotting character, of your choice, and a nice color (see pch and col in the help for par). Note that pch and col can be used directly as parameters in the plot function. Use the function abline to add a vertical line at the lead concentration value 1.0 in the plot produced. If you feel creative, improve the plot in other ways.
Simulation in R

We will now see how simulation can be used to learn about statistical issues and solve concrete problems.

1. We will use as examples 4 distributions which have most or all of their probability density in the interval between 0 and 10:
   - The Normal distribution with mean 5 and standard deviation 2.
   - The Uniform distribution on the interval from 0 to 10.
   - The Chi-Square distribution with 3 degrees of freedom.
   - The F distribution with 20 and 3 as the first and second degrees of freedom, respectively.

Start with making 4 plots, one for each distribution, showing the probability density function on the interval from 0 to 10. Start by creating the x-vector with numbers from 0 to 10, evenly spaced with some small number as step size. Use the help system to investigate the functions `dnorm`, `dunif`, `dchisq` and `df` to get the y-values needed for each plot. You can place all plots in the same figure by executing the command

```r
> par(mfrow=c(2,2))
```

before the four plot commands.

2. For each of the 4 distributions, simulate 10000 random values using the functions `rnorm`, `runif`, `rchisq` and `rf`. Save the simulated values in 4 vectors and visualize them in histograms (Use `breaks='Scott'` as a parameter to get more detail in the histograms). How do the histograms compare to the density functions plotted earlier?

3. For each of the 4 distributions, use the functions `pnorm`, `punif` etc. to compute the probability of observing a value above 10. Then, obtain approximations of these probabilities using simulation. For each distribution, simulate 10000 values and find which proportion of these values that are above 10. Run the simulations a few times for each distribution to get an impression of the variability of the result.

4. Emma is performing an experiment that requires individual handling of some animals. The sizes of the animals are lognormally distributed: The natural logarithms of their sizes has a normal distribution with mean 3 and standard deviation 0.4. Simulate a vector `s` with the sizes of 10000 animals. In general, a lognormal variable can be simulated by first using `rnorm` and then using the function `exp` on the result.

The time in minutes it takes to handle each animal is given by

\[
10 + s \cdot 1.5 + e^\epsilon \quad \text{for animals with } s \leq 20, \\
20 + s \cdot 0.8 + e^\epsilon \quad \text{for animals with } s > 20,
\]

where \( \epsilon \) is a random variable that is normally distributed with expectation 1 and variance 0.3. For a randomly picked animal, what is the probability that it can be handled in less than 30 minutes? Find an approximate answer to this question using simulation.
5. The following theoretical results hold in general:

\[ X \sim \text{Normal}(\mu, \sigma) \implies E(X) = \mu, \quad \text{Var}(X) = \sigma^2, \]
\[ X \sim \text{Uniform}(a, b) \implies E(X) = \frac{a + b}{2}, \quad \text{Var}(X) = \frac{(b - a)^2}{12}, \]
\[ X \sim \text{Chi-Square}(d) \implies E(X) = d, \quad \text{Var}(X) = 2d, \]
\[ X \sim F(d_1, d_2) \implies E(X) = \frac{d_2}{d_2 - 2} (\text{for } d_2 > 2). \]

For each of the 4 distributions, with parameter values as specified in the first step, simulate 10000 values and compute their mean and variance (using the R functions `mean` and `var`). Compare the values obtained with the theoretical mean and variance. How good is the approximation? Repeat the simulation a couple of times, to get an impression of the variability of the results. Note that for the F distribution in this case \( d_2 = 3 \), the variance does not exist.

6. The procedure performed in the previous step estimates the true mean and variance from a random sample of size 10000. In real life, we usually have much smaller sample sizes when trying to estimate the (unknown) mean. In the next step you will simulate random samples of different sizes and learn/recall how the estimate of the mean behaves and depends on the sample size.

Consider the following function:

```r
simulate.chisq.means <- function(k, n) {
  x <- rchisq(k * n, df = 3)
  M <- matrix(x, nrow = k, ncol = n)
  means <- apply(M, 1, mean)
  return(means)
}
```

Given values for \( k \) and \( n \), the function returns a vector of length \( k \). Each entry of this vector constitutes an independent realisation of the sample mean for a random sample of size \( n \) taken from the Chi-Square distribution with 3 degrees of freedom. Use this function to simulate 10000 sample means, each corresponding to a sample of size 3. Make a histogram of the simulated values. Then, make three similar histograms, but where you take the mean over 10 values, 30 values, and finally 100 values, respectively.

7. Make three copies of the function in the previous step and modify them so that they can be used to generate vectors of sample means for the other three distributions specified in the first step. Now, for each of the four distributions considered, do the following: generate sample mean vectors of size \( k = 10000 \) for \( n = 1, 3, 10, 30, 100 \) and compute the sample variance for each one of these five vectors. Plot the sample variances against the values of \( n \). Do you see a pattern? Can you formulate precisely what the relationship between \( n \), the variance of the distribution and the resulting variance for the mean is? Is the pattern the same for all the distributions?

**Quantiles and Q-Q plots**

We will study the dataset `stroke` which is part of the ISwR package (the companion package to our textbook “Introductory Statistics with R”). Make sure you have installed the package ISwR;
in R Studio you may use the “Install packages...” command under the “tools” menu. In any R environment you can also use the command,

```r
> install.packages("ISwR")
```

You might need to select your CRAN mirror using `chooseCRANmirror()`. Activate the package with the command

```r
> library(ISwR)
```

Check out the contents of the package with the command

```r
> help(package=ISwR)
```

and load the dataset `stroke` from it with the command

```r
> data(stroke)
```

Check with `ls()` that you now have an object called `stroke` in your workspace. Get an overview of what this object contains by applying the functions `help`, `summary`, and `head` to it.

1. We will look at how the age at stroke varies: Assign this column of the `stroke` dataset to a vector called `Age`. Use the function `quantile` to find the quartiles and the deciles of the distribution of ages. Illustrate the data with a histogram using `hist` and with a curve showing the empirical cumulative distribution function using `ecdf`.

2. Are the ages normally distributed? Use the `qqnorm` function (and possibly also `qqline`) on `Age` to investigate this question, and try to interpret the result.

**Part 2**

The second part of today’s lab is concerned with applying some basic bootstrap techniques for data analysis. In order to have a convenient setting for understanding the performance of the different techniques we will not use a real data set, but instead simulate random samples from a gamma distribution. To indicate that a random variable $X$ follows a gamma distribution with shape parameter $\alpha > 0$ and rate parameter $\beta > 0$, the notation $X \sim \text{Gamma}(\alpha, \beta)$ is used. The probability density function for such a random variable has the form

$$f_X(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}, \quad x \geq 0,$$

and the theoretical mean and variance of $X$ may be expressed in terms of the parameters according to

$$E(X) = \frac{\alpha}{\beta}, \quad \text{Var}(X) = \frac{\alpha}{\beta^2}.$$

Given a random sample $X_1, \ldots, X_n \sim \text{Gamma}(\alpha, \beta)$ of independent observations, our focus will be on investigating the properties of the sample mean

$$\hat{\theta} = \frac{1}{n} \sum_{i=1}^{n} X_i$$

as an estimator for the true mean $\theta = \theta(\alpha, \beta) = \alpha/\beta$. It may be shown that

$$\hat{\theta} \sim \text{Gamma}(n\alpha, n\beta).$$
1. Consider the function `non.parametric.bootstrap`:

```r
non.parametric.bootstrap <- function(s, k) {
  means <- vector(mode = "numeric", length = k)
  for (i in 1:k) {
    resample <- sample(s, length(s), replace = TRUE)
    means[i] <- mean(resample)
  }
  return(means)
}
```

Given a vector of numbers `s` drawn from some distribution, this function draws `k` resamples and computes and returns a vector of sample means. Use `rgamma` with shape parameter \( \alpha = 2 \) and rate parameter \( \beta = 1 \) to construct a random sample `s` of size 100. Then, with the help of `non.parametric.bootstrap`, draw 1000 resamples and illustrate the bootstrap distribution of \( \hat{\theta} \) in a histogram. What are the estimated bias and variance of \( \hat{\theta} \)? How do the values obtained compare to the theoretical ones? Hint: It is helpful to plot the theoretical density of the sample mean in the same figure as the histogram, using `dgamma` with shape parameter 100\( \alpha \) and rate parameter 100\( \beta \).

2. The non-parametric bootstrap procedure makes no assumptions about the distribution from which the sample `s` originates. Modify `non.parametric.bootstrap` to obtain a new function (name it `parametric.bootstrap`) that assumes `s` comes from a gamma distribution. Before resampling begins in `parametric.bootstrap`, \( \alpha \) and \( \beta \) need to be estimated. This can be done numerically using maximum likelihood via the function `MASS::fitdistr`. Starting with a sample `s` of the same form as in the previous step and using your new function, draw 1000 resamples and illustrate the bootstrap distribution of \( \hat{\theta} \) in a histogram. Do you get reasonable values for the estimated bias and variance of \( \hat{\theta} \)?

3. Use either the parametric or non-parametric bootstrap distribution based on a sample `s` of size 100 to construct 95% confidence intervals (CIs) for \( \theta \). Try and compare each of the three methods covered in the lecture: basic CI, normal CI and percentile CI. For constructing the basic and percentile CIs the function `quantile` is useful.

4. Unless you are faced with a very special problem, you will probably use the existing package `boot` when solving bootstrap problems using R in the future. Take a few minutes and use the help system to find out more about the functions `boot` and `boot.ci`. Use the function `boot` to perform a non-parametric bootstrap analysis of a gamma sample `s`, construct CIs from the output using `boot.ci` and check the results against those obtained using your own functions.

5. Bootstrap is known to give less reliable results if the sample size `n` of `s` is small. Study this by doing the following. First, let `n = 10` and draw 1000 samples of this size. For each sample, calculate a basic 95% CI and record whether it covers the true value of \( \theta \). Repeat this for `n = 20, 30, \ldots, 100`. For each `n`, does the CI have the correct coverage probability? Present the results in a graph. You can use either your own code or `boot.ci` to get the bootstrap CIs.

6. (OPTIONAL, do this if you want a challenge) Repeat the previous step, but let the parameter to be estimated be the variance \( \eta(\alpha, \beta) = \alpha / \beta^2 \) instead of the mean of the gamma distribution. What can be said about the performance of this confidence interval? What do you think is the reason? Next, construct studentized confidence intervals for the variance and check the coverage for different sample sizes. Any improvement?