Demo 2: Descriptive statistics and plotting the data

Experimental and Statistical Methods in Biological Sciences I

September 25, 2014

1 Load in and prepare AgeWeight data

Load in the data for the exercises: the AgeWeight data from last week. It can be found at `http://becs.aalto.fi/~heikkih3/age_weight2.csv`. Save the data as dataframe `ageweight`. Remember `setwd, read.csv`.

Examine you data using `head, summary`.

Is there anything you need to change? Check that factors are factors, and check for missing values. You potentially need `factor, summary, which`.

Note a new thing: no need to overwrite your data. Save changes to a NEW dataframe, e.g. ageweight2, ageweight3, ... or whatever name you like. You can then review and see what kind of impact your changes had.

```
# Save your original dataframe:
> ageweight.orig <- ageweight
# Then back to the ageweight data.
summary(ageweight)
# MALE is again not a factor. Let's change it:
> ageweight$MALE <- factor(ageweight$MALE)
# WEIGHT has weird values (-9??)
> which(ageweight$WEIGHT <= 0)
[1] 21 36 95
# Now code these as missing values to WEIGHT:
> ageweight[which(ageweight$WEIGHT <= 0), "WEIGHT"] <- NA
```

Happy now with your dataframe? If so, use `attach` to attach the `ageweight` data so that you don't have to always use `$` to assign variables.

2 Basics in plotting

How to build your plots in R:

1. Use a plotting function (e.g., `plot, hist, boxplot`) to generate the type of plot you want;
2. Boost your plot by adding named arguments that modify your plot;
3. Save your plot.

2.1 Basic structure of plotting functions

There are different functions for different kinds of plots, e.g. `plot, hist, boxplot, pie`. Sometimes, the type of the plot you get when calling a certain function for a certain variable also depends on the data type of the variable. Test for example the function `plot`:

```
# For a numeric variable:
> plot(WEIGHT)      # a scatterplot
# For a factor:
> plot(MALE)        # a bar chart
```

You can also include several variables in `plot`:

```
# Two numeric variables:
> plot(AGE, WEIGHT)    # a scatterplot
> plot(WEIGHT~AGE)     # does the same thing
                       # ~ refers to 'by', e.g. Weight by Age
                       # i.e., Weight as a function of Age
# A factor and a numeric variable:
> plot(MALE, WEIGHT)   # a box plot for categories separately
                       # notice that the order matters!
# Two factors:
> plot(MALE, SMOKE1)   # a bar chart
```

You can make more specific kinds of plots using different functions:

```
# Histograms (numeric variables only):
> hist(WEIGHT)


# Box plots (numeric variables only):
> boxplot(WEIGHT)
```

2.2 Modifying your plots

As you saw, the plotting commands create a very basic form of a plot. To boost this plot, you have to add named arguments to your functions. Here are some examples for function `plot`:

```
# Add main title
> plot(WEIGHT~AGE, main="Weight by Age")
# Add/modify axis labels
> plot(WEIGHT~AGE, main="Weight by Age", xlab="Age (yrs)", ylab="Weight (kg)")
# Change point style
> plot(WEIGHT~AGE, main="Weight by Age", xlab="Age (yrs)", ylab="Weight (kg)", pch=16)
# Change point color
> plot(WEIGHT~AGE, main="Weight by Age", xlab="Age (yrs)", ylab="Weight (kg)", pch=16, col="blue")
# Adjust axes
> plot(WEIGHT~AGE, main="Weight by Age", xlab="Age (yrs)", ylab="Weight (kg)", pch=16, col="blue",
xlim=c(0,70), ylim=c(0,140))
```

You can also add other graphical features in your plot, e.g., a line:

```
# Add a line, here, a line that shows the best linear fit:
> plot(WEIGHT~AGE, main="Weight by Age", xlab="Age (yrs)", ylab="Weight (kg)", pch=16, col="blue")
> abline(lm(WEIGHT~AGE))
# You can also modify the line:
> abline(lm(WEIGHT~AGE), lty="dashed", lwd=2, col="red")
```

Obviously, other types of plots can be modified as well:

```
# Make a nice histogram:
> hist(WEIGHT, main="Weight", xlab="Weight", ylab="")
# You can also set the total area size to 1:
> hist(WEIGHT, main="Weight", xlab="Weight", ylab="", freq=F)
```

Notice that the function `hist` sets the bin size automatically. You might want to change the bin size. There are two ways to do this:

```
# Automatically selected bin size:
> hist(WEIGHT)
# You can either suggest the number of bins...
> hist(WEIGHT, breaks=10)
# ... Or the size of a bin.
> bins = seq(20,140,by=20)   # create bins of width 20kg each
> hist(WEIGHT, breaks=bins)
```

Sometimes you might want to add a normal curve approximation too. We need to create and put together a few pieces in order to do this...

```
# Create the histogram object first
> h <- hist(WEIGHT, freq=F, main="Distribution of Weights", xlab="Weight (kilos)", ylab="")
# Find highest and lowest x values and create a sequence of them
> x <- min(h$breaks):max(h$breaks)
# Create vector y from normal distribution with same mean and sd as WEIGHT
> y <- dnorm(x, mean=mean(WEIGHT, na.rm=T), sd=sd(WEIGHT, na.rm=T))
# Finally, add the normal curve to your histogram:
> lines(x,y,col="red", lty="dashed", lwd=3)
```

2.3 Opening several plots in the same display

If you want to include multiple plots at one display, use `layout` command before running any plot commands. Try this with different matrices:

```
# E.g., put 4 plots in the same display:
> layout(matrix(c(1,2,3,4),2,2))
> hist(WEIGHT, main="Weight")
> hist(HEIGHT, main="Height")
> plot(SMOKE1, main="Smoking at time point 1")
> plot(SMOKE2, main="Smoking at time point 2")
```

Tip: closing the graphics window will set `layout` back to default.

2.4 Saving your plots

To save your plot, you need to first call a device to which your plot is saved (default is the graphics display), then create your plot, and finally close the device:

```
> pdf(file="hist_weight.pdf")
> hist(WEIGHT, main="Weight", xlab="Weight (kg)", ylab="")
> dev.off()
```

You should now be able to find a file called 'hist_weight.pdf' which includes your graph in your working directory. There are many formats in which you can save your graphs, including .ps, .png, .jpeg, ... You can check the full list by running `?Devices`.

2.5 Exercises

Question 1 Try

```
> hist(WEIGHT, xlim=c(0,150), breaks=6, col="red", xlab="Weight (kg)", bg="grey")
```

What do the various parameters do? Tip: to find out more about the parameters, try `?hist` and `?par`.

Question 2 Find a way to plot the same histogram without its main title.

Question 3 How do you think weight will vary with age? Try `plot(AGE,WEIGHT)`. Were you right?

Question 4 Now redo the scatterplot using `plot(AGE,WEIGHT)` and adding named arguments. Experiment with `pch` ('plot character') values other than 16, make the points colored. You can also clarify your results by adding a regression line with `abline` (see example in Section 2.2).

Question 5 We just saw how weight varies with age - but does weight vary with gender? Check out the relationship between these two variables by plotting.

Question 6 Why does `plot` produce different graphs in each case (in Questions 3 and 5)?

Question 7 Find out how to change the font in your graphs. You might find help in `www.statsmethods.net/advgraphs/parameters.html`.

3 Descriptive statistics & plotting of categorical variables

3.1 Contingency tables

A contingency table is an effective way of presenting frequencies of different categories.

With `table`, you can create all kinds of frequency tables. Let's start with just one categorical variable:

```
> table(MALE)
```

This is not yet very useful, because we could also get the same information using `summary`. However, things get more interesting when you build a table with multiple categorical variables - this is also called a contingency table. We can start by examining whether it seems that more women or men smoke. Let's make a contingency table:

```
> table(MALE, SMOKE1)
```

We can also express the frequencies as percentages of women and men who smoke:

```
> round(prop.table(table(MALE, SMOKE1), margin=1)*100, 1)
```

It seems that more women than men smoke in our dataset. In the above command, pay careful attention to the brackets when working out what's going on: for example, you can see that the command is really `round(x,1)` where `x` is `(prop.table(...)*100`.

3.2 Plotting

You can take bar plots of categorical variables simply by using `plot`:

```
> plot(MALE)
```

We can also boost this simple plot:

```
> plot(MALE, col=c("red", "blue"), xlab="Gender")
```

This kind of plotting works also for two categorical variables:

```
> plot(MALE, SMOKE1)
> plot(MALE, SMOKE1, col=c("red", "blue"), xlab="Gender", ylab="Smoking")
```

Adding this to your publication or presentation is not very informative, so we prefer either contingency tables or simply descriptive statistics embedded in text.

3.3 Exercises

Question 8 How many people have quit smoking between time points 1 and 2? Tip: Take a contingency table of `SMOKE1, SMOKE2` and think through how to interpret the different cells.

Question 9 Plot the table from Question 8. Modify your plot so that it displays the information efficiently.

4 Descriptive statistics & plotting of continuous variables

4.1 Descriptive statistics

You can see a summary of the basic descriptive statistics (including range, mean, 1st and 3rd quartiles) using summary. Another very useful function is `describe` (from library `psych`). `describe` shows e.g. sample size, mean, sd, meadian, range, skewness, kurtosis, ... for all the variables in the data frame.

Let's try this:

```
> summary(ageweight)
> install.packages('psych')
> library('psych')
> describe(ageweight)
# You can also take each descriptive statistic separately:
> mean(WEIGHT, na.rm=T) # mean
> var(WEIGHT, na.rm=T) # variance
```

Note the use of `na.rm=T` (you can try without!) - this removes the missing values when calculating the statistics.

It is also useful to examine the descriptive statistics for categorical variables separately. For instance, we might want to take the means of `WEIGHT` for men and women separately. An efficient way to do this it the very useful `tapply` function. With `tapply`, you can calculate any function (e.g., mean, sum, ...) of a cross-tabulation of data. The first argument is a vector of outcome variables. The second argument is a list of one or more factors that is used to break the first argument into smaller pieces. The third argument is a function that is applied to each piece of data:

```
> tapply(WEIGHT, MALE, mean, na.rm=T)
```

Note that you can also use `tapply` for multiple categorical variables. Here we show the average weights for male and female smokers and non-smokers:

```
> tapply(WEIGHT, list(MALE, SMOKE1), mean, na.rm=T)
```

Saving your tables

You can also save your tables in a format that can be later read and modified in some other software (e.g., Excel):

```
> my.table <- tapply(WEIGHT, list(MALE, SMOKE1), mean, na.rm=T)
> write.csv(my.table, "my_table.csv")
```

4.2 Plotting

We have already seen many of the useful plot commands for creating plots for numeric variables. The two most important ones are `boxplot` and `hist`. If you want to learn how to produce all the plot types introduced in the lecture, check out `www.harding.edu/fmccown/r`.

To display the characteristics of a numeric variable in a concise manner, use a box plot:

```
> boxplot(WEIGHT)
```

The box plot shows you the median, the upper and lower quartiles, and where 90% of the observations lie (if you need a recap, see lecture slides).

Note how a box plot can be used to spot outliers in the data. Try `boxplot` for both `ageweight$WEIGHT` and `ageweight.orig$WEIGHT`:

```
> layout(matrix(c(1,2)))
> boxplot(ageweight$WEIGHT)
> boxplot(ageweight.orig$WEIGHT)
```

Also, boxplots can be an efficient way to compare the distributions of a variable in different categories. When using `boxplot` this way, a useful rule of thumb is to think that you are plotting numeric variable BY factor:

```
# e.g., plot WEIGHT by MALE (gender)
> boxplot(WEIGHT~MALE)
```

To further examine the distribution of a variable, use `hist`:

```
> hist(WEIGHT)
> hist(WEIGHT, breaks=20))
```

Remember to modify your graphs if you want them to be publication-quality!

4.3 Exercises

Question 10 What is the mean and standard deviation of `HEIGHT` in the whole data set?

Question 11 What is the mean and standard deviation of `HEIGHT` for (a) women, (b) men? Present the values in a table and save as "Height_by_gender.csv".

Question 12 Plot a histogram of `HEIGHT`.

Question 13 Add a normal distribution approximation to your histogram of `HEIGHT`. Tip: go back to section 2.2 to see how this is done.

Question 14 Try modifying the plot with R. E.g., change colouring, increase bin size, change axes, add labels.

Question 15 Save your final plot in .pdf format.

Question 16 Plot the distribution of height as a boxplot this time. Using the plot, what can you infer about the (a) central tendency, (b) dispersion of the variable `HEIGHT`?

Question 17 Plot the distribution of height in men and women separately using a boxplot. Are there differences in height between men and women according to the plot?

Question 18 Plot the distribution of `WEIGHT` in (a) one boxplot, and (b) one histogram, displayed in the same graphics window (tip: see section 2.3). What is the best way to present the distributions? Why? How do the results look like?

Question 19 Modify your winning plot from the previous question into a publication-quality graph. Once you are happy with how the graph looks like, save it in a .pdf file.

5 Exercises

Try descriptive statistics and plotting with the naming dataset from last week. You can find the dataset here: `http://becs.aalto.fi/~heikkih3/naming_new.csv`.

Load the dataset into a data frame in R using `read.csv`. Remember to check for factors and missing values first! Correct these if necessary using the guidelines from last week.

Once your data frame is in its final format, it is useful to attach it. First, remove `ageweight` dataset by using `detach(ageweight)`, then attach naming instead (`attach(naming)`).

Question 20 Do reading times differ between the different word types (exception and regular)? Does gender have an effect in these differences? Examine plots. Useful functions: `plot, tapply`.

```
# Some tips:
> plot(word.type, ms)
> layout(matrix(c(1,2)))
> plot(word.type, ms, data=naming[which(sex="female"),], main="Female")
> plot(word.type, ms, data=naming[which(sex="male"),], main="Male")
```

Question 21 Create a subset of the data where you include only the background variables (`iq, hrs, sex`) from naming data frame and only one observation from one participant. (Remember the data structure: rows 1-1000 contain same participants as rows 1001-2000!) Save this subset as `naming_subset`.

From now on, we will use only the `naming_subset` data frame. Therefore, detach `naming` and attach `naming_subset`.

Question 22 Create a factor 'reading_group' based on hours spent reading:

- if hours $=<$ 3.5, low
- if hours $>$ 3.5 but $=<$ 4.5, medium
- if hours $>$ 4.5, high

```
# Example:
> naming_subset$reading.group <- 'low'
> naming_subset$reading.group[which(hrs > 3.5)] <- 'medium'
> naming_subset$reading.group[which(hrs > 4.5)] <- 'high'
> reading.group <- factor(reading.group)
> attach_subset(naming)
```

Question 23 Make a contingency table of gender and reading group. Judging by the numbers, are there differences between men and women in the proportion of different reading time categories?

```
# Example:
> table(sex, reading.group)
> plot(table(sex, reading.group))
```

Question 24 Next, observe the same but this time by plotting the numeric variable hrs with gender (i.e., plotting the hrs separately for men and women).

```
# Example:
> boxplot(hrs~sex))
```

Question 25 Plot intelligence separately for men and women. Do you see differences?

Question 26 Plot the histogram of intelligence. Does the distribution follow the normal curve? Tip: try adding a normal curve to the plot. You can also read through section 6 and try examining QQ-plot and testing for normality.

Question 27 Collect the descriptive statistics for this dataset into publication quality table(s). Plan this ahead! Note at least:

- `naming` includes background variables for each participant twice, `naming_subset` once - which data frame should you use?

- you find the results for the experimental manipulation `word.type` in `naming` data frame

- it might be a good idea to put background variables in one table and results from experimental manipulation in another, or what do you think?

Save your table as .csv file and use Excel (or something alike) to modify your table into a publication-quality table.

6 Basic tools for testing for normality

Parametric statistical tests assume that the data come from population that follows normal distribution. There are several methods for checking the normality. The basic tools include plotting the distribution (histograms, QQ-plots) and testing for normality.

6.1 Histogram

We prepare data for this exercise by using `rand` function to generate a set of random numbers.

First, let's take a sample of 20 from the normal distribution with mean = 0 and sd = 1 (function `rnorm` uses these parameters by default):

```
> set.seed(111)
> data_normal <- rnorm(20))
```

We will also prepare another kind of data for comparison, namely, data that follows an exponential distribution:

```
> data_exp <- rexp(20)
```

First thing before doing any statistical test for normality is to look at the histogram of the data. This gives you a good picture of what the data look like. In R, use `hist` function to create a histogram. To make the comparison easier, put the two histograms into one window:

```
> layout(matrix(c(1,2), nrow=1))
> hist(data_normal)
> hist(data_exp)
```

6.2 Q-Q plot

Another way to visually investigate whether data follow the normal distribution is to draw a Q-Q ('quantile-quantile') plot. A Q-Q plot shows the mapping between the distribution of the data and the ideal distribution (the normal distribution in this case). Let's take a look at it.

```
> layout(matrix(c(1,2), nrow=1))
> qqnorm(data_normal)
> qqline(data_normal)
> qqnorm(data_exp)
> qqline(data_exp)
```

If your data are close to the normal distribution, most of the data points should be close to the line.

6.3 Statistical tests for normality

If the histogram of your data doesn't really look like a normal distribution, you should try a statistical test to check for the normality. Fortunately, this is pretty easy in R. There are two common tests you can use for the normality check:

Shapiro-Wilk test

This works well even for a small sample size, so generally you just need to use this. The null hypothesis of Shapiro-Wilk test is that the samples are taken from a normal distribution. So, if the p value is less than 0.05, you reject the hypothesis, and think that the samples are not taken from a normal distribution. In R, you just need to use `shapiro.test()` function to run the Shapiro-Wilk test.

```
> shapiro.test(data_normal)
> shapiro.test(data_exp)
```

Kolmogorov-Smirnov test

For larger sample sizes. See more: `yatani.jp/HCIstats/DataTransformation`.

```
> ks.test(data_normal, "pnorm", mean=mean(data_normal), sd=sd(data_normal))
> ks.test(data_exp, "pnorm", mean=mean(data_exp), sd=sd(data_exp))
```