Untitled

# Advanced Statistics

author: Bernhard Angele date: Class 3, October 16, 2014

# Your last serving of R basics

* Working with files
* R can open all kinds of data files
* You do have to tell it where to find them
* R's working directory
* This is where R looks for files to open
* You change it using `setwd("C:/My\_example\_R\_directory")
* No backslashes \ allowed! Use forward slashes instead /

# Setting your working directory

* Don't know where your working directory should be?
* Start by putting your script files in a location that is sensible (to you)
  + e.g. C:\Some\_Document\_Folder\AdvStats\Class3\ or /home/My\_Name/Documents/MastersProject/Data
  + That directory should also be your working directory for the current session
  + In RStudio, open your script file, then use the menu bar and go to Session --> Set Working Directory --> To Source File Location
  + RStudio will write a setwd command to the console.
  + Copy and paste this command to your script, and you're set. Every time you run the script, the working directory will be automatically set
* When you use R Markdown, Knitr sets your working directory to the location of the Rmd file automatically.

# Opening files and importing data

* If you have data in Excel:
* Save them as comma-separated values (CSV)
* If you are using Excel formulas and other bells and whistles, check that your data are exported correctly
* Make sure the first row of every column contains its column name
* Save this file in your working directory (remember where that was?)
* Now you can import the file using new\_object\_with\_a\_sensible\_name <- read.csv(file = "my\_filename.csv")
* If there is an error/warning message, check that there are no missing cells
  + Same number of data points in each row

# Data Frames

* A Data Frame is a combination of a list and a matrix
* It gets its row/column structure from the matrix
* But you can use columns of any data type (numeric, logical, factor (discrete), logical, character)
* Perfect for the kind of analyses we would like to do
* Very similar to SPSS data files
* Get a column by using object\_name$column\_name
* Get a column by slicing: object\_name[,1]
* Get a row by slicing: object\_name[1,]
* Get a cell by slicing: object\_name[1,1]

# Packages

* Many authors have worked on extending R and teaching it new tricks
* Packages are collections of functions that help you perform a certain task
* For example, the ez package has all kinds of useful functions for performing ANOVAs
* To install a package, type install.packages("package\_name") at the Console.
* e.g. install.packages("ez")
* You can ask R to install multiple packages at once:
* Run this line in the console now: `install.packages(c("ez","reshape","ggplot2","lme4", "plyr"))

# Loading packages

* Installed packages are not loaded automatically
* If you want to use a command from a package, you need to load it: library(package\_name)
* e.g. library(ez)
* Once it is loaded, the package and all of its commands are ready to go until you restart R.

# Starting a script

Good practice when starting a script:

# My\_testscript.R  
# Author: Bernhard Angele  
# Description: Demonstrates how to start a script  
###########################  
  
rm(list = ls()) # clear the workspace to avoid problems caused by old objects  
library(ez) # load (only) those libraries that you need  
library(reshape)  
library(ggplot2)  
# Set your working directory  
setwd("C:/I\_know\_my\_working\_directory\_and\_this\_is\_it/")  
# Now load your data, do your analyses and win fame and fortune!

# Starting an Rmd file

* Fill in the header:

---  
title: "My fantastic results section"  
output: word\_document  
---

* Then do the same things as above in the first code chunk
* Hint: you can tell R to hide this boring code by adding hide = TRUE to the header. It will still get evaluated

Start your code chunk like this: ```{r, hide = TRUE}

# Comparing multiple groups

* t-tests are nice if you only have two groups that you want to compare.
* But maybe you have more groups
* Example: > A researcher wants to find out if there is a systematic difference in intelligence between MSc students from different universities. She performs intelligence tests on 10 students each from BU, University of Southampton and Oxford University and records the results.

# Making fake data for our example

* Let's assume that the true state of affairs is that there is no difference in intelligence
* In that case, all intelligence scores would come from the same distribution: a normal distribution with mean = 100 and sd = 15
* Let's generate 3 data sets according to this criterion:

bu <- rnorm(n = 10, mean = 100, sd = 15)  
soton <- rnorm(n = 10, mean = 100, sd = 15)  
oxford <- rnorm(n = 10, mean = 100, sd = 15)

# What if we use t-tests?

* We could simply perform t-tests on these data
* How many would we need?
* 3: BU vs. Soton, BU vs. Oxford, Soton vs. Oxford

# BU vs Soton

t.test(bu, soton)

##   
## Welch Two Sample t-test  
##   
## data: bu and soton  
## t = -0.459, df = 17.7, p-value = 0.6518  
## alternative hypothesis: true difference in means is not equal to 0  
## 95 percent confidence interval:  
## -17.13 10.99  
## sample estimates:  
## mean of x mean of y   
## 95.05 98.12

# BU vs Oxford

t.test(bu, oxford)

##   
## Welch Two Sample t-test  
##   
## data: bu and oxford  
## t = 0.5802, df = 17.61, p-value = 0.5691  
## alternative hypothesis: true difference in means is not equal to 0  
## 95 percent confidence interval:  
## -11.75 20.69  
## sample estimates:  
## mean of x mean of y   
## 95.05 90.58

# Soton vs Oxford

t.test(soton, oxford)

##   
## Welch Two Sample t-test  
##   
## data: soton and oxford  
## t = 1.03, df = 16.73, p-value = 0.3175  
## alternative hypothesis: true difference in means is not equal to 0  
## 95 percent confidence interval:  
## -7.919 23.001  
## sample estimates:  
## mean of x mean of y   
## 98.12 90.58

# Anything wrong with that?

* We are doing three independent t-tests
* Each t-test has a 5% chance of producing a spurious result ()
* What is the chance that we get at least one spuriously significant result?
* It's 1 - the chance that we get no spurious results
* We have a problem: our is almost three times as high as it should be.
* SPSS calls this LSD (least significant differences) - don't use it!

# Exercise: Give it a try

* Instead of running the simulation on my computer, I'll run it on **you**
* What I mean by that:
* Generate three data sets like I've just shown you
* Make sure that all three data sets are samples from the same normal distribution
  + Same mean and sd, of course same n as well
* Run three two-sample t-tests comparing the means
* Afterwards, I will ask you and count how many of you found at least one significant result.
* If the level isn't inflated, only one or two of you should find one

# Solutions

* We can adjust the level of each t-test:
* If we divide the alpha level by the number of tests, we get
* Our total is then:
* This is called a **Bonferroni correction**
* Problem solved?
* Yes, but the lower the level, the lower the power.
* Better ways (but still lowering power):
* Tukey's HSD (honestly significant differences)
* Scheffe's test
* Maybe we just want to know if there is a difference at all between these three means
* One-way ANOVA

# Putting our fake data in a different format

* Usually, we aren't getting our data neatly in different objects
* Instead, we'll have a big table with our data
* If we are going to put our data in this format, we need a variable (or column) identifying the group
* Let's add that to each of our data sets

bu <- data.frame(iq = bu, group = "BU")  
soton <- data.frame(iq = soton, group = "Soton")  
oxford <- data.frame(iq = oxford, group = "Oxford")

# Looking at these data now

bu

## iq group  
## 1 105.62 BU  
## 2 65.47 BU  
## 3 100.11 BU  
## 4 103.56 BU  
## 5 109.78 BU  
## 6 111.61 BU  
## 7 100.26 BU  
## 8 73.59 BU  
## 9 99.32 BU  
## 10 81.18 BU

# Putting it all together

iqdata <- rbind(bu, soton, oxford)  
str(iqdata)

## 'data.frame': 30 obs. of 2 variables:  
## $ iq : num 105.6 65.5 100.1 103.6 109.8 ...  
## $ group: Factor w/ 3 levels "BU","Soton","Oxford": 1 1 1 1 1 1 1 1 1 1 ...

* Data frames are smart:
* R has automatically converted group into a factor
  + A factor is a discrete variable (nominal scale)
* R left iq as a numeric variable
  + Numeric variables are continuous (interval or ratio scale)

# Running an ANOVA -- by hand!

* One-way ANOVAs are so simple, all you need is a calculator (or R, even better)
* First, we need an estimate of the total variance in the data: calculate the total sum of squared deviations from the mean (or short, sum of squares, SS)
* , where is each individual value and is the grand mean of the data.

(SS\_tot <- sum((iqdata$iq - mean(iqdata$iq))^2))

## [1] 7383

# Running an ANOVA -- by hand! (2)

* Now we need an estimate of the variance explained by group
* , where denotes the mean for each group, is the number of subjects in each group, and is the grand mean of the data.

bu\_mean <- mean(subset(iqdata, group == "BU")$iq)  
soton\_mean <- mean(subset(iqdata, group == "Soton")$iq)  
oxford\_mean <- mean(subset(iqdata, group == "Oxford")$iq)  
n <- 10  
  
(SS\_model <- sum(n\*(bu\_mean - mean(iqdata$iq))^2, n\*(soton\_mean - mean(iqdata$iq))^2, n\*(oxford\_mean - mean(iqdata$iq))^2))

## [1] 287.6

# Running an ANOVA -- by hand! (3)

* Now get an estimate of the variance that is *not* explained by group, i.e. the error.
* Easy: Just subtract from .

(SS\_error <- SS\_tot - SS\_model)

## [1] 7095

# Running an ANOVA -- by hand! (4)

* With and , we can compute the ratio of explained variance to error (or unexplained) variance.
* First, we need to take into account the number of measurements which went into each SS
* These are called the degrees of freedom
* This is equivalent to calculating the variance as a descriptive statistic ()
* , where is the number of observations per group (10) and is the number of groups
* Why -1? Very good question. Short answer: Whenever we want to make conclusions about the population, we use n-1 instead of n.
* Long answer (if you are REALLY interested and won't shut up about it): [Here](http://nebula.deanza.edu/~bloom/Math10/M10DivideBy_nminus1.pdf)

# Running an ANOVA -- by hand! (5)

* , where is the number of groups.

k <- 3  
(df\_total <- n \* k - 1)

## [1] 29

(df\_model <- k - 1)

## [1] 2

(df\_error <- n \* k - k)

## [1] 27

# Running an ANOVA -- almost done!

* Now we compute the mean squares (MS) as an estimate of the variance

(MS\_model <- SS\_model/df\_model)

## [1] 143.8

(MS\_error <- SS\_error/df\_error)

## [1] 262.8

# Running an ANOVA -- final steps!

* Finally, we take the ratio of the two.

(F\_value <- MS\_model/MS\_error)

## [1] 0.5473

# What to do with this F-value

* It turns out that the ratio between model and error variance follows a specific distribution
* If there is no actual effect (!) and
* As long as certain assumptions are valid (more on that later)
* This distribution is called the F-distribution
* Occasionally you will get a high simply by chance, but such occurrences are quite rare
* The F-distribution is the probability density function for different values of the variance ratio, i.e. the F-value.
* We essentially want to test if the F-value we get is extreme enough that it could only have occurred by chance 5% of the time (our level)

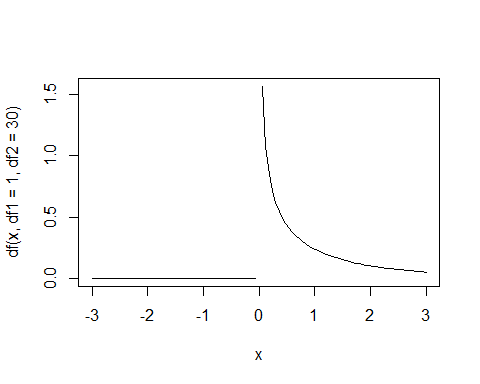
# The F-distribution

* Like the *t*-distribution, the shape of the F-distribution varies depending on sample size (degrees of freedom).
* Remember that F is a *ratio* of two variances.
* Because of this, the F distribtion has *two* degrees of freedom parameters
* Guess what the R functions for the F distribution are called
* That's right: df, pf, qf, rf

# Plotting it

* Let's take a look

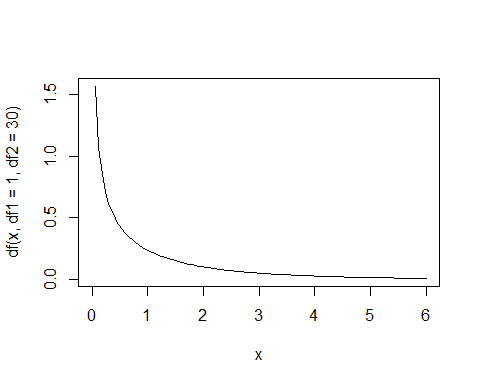
curve(df(x, df1 = 1, df2 = 30), from = -3, to = 3)

 - Looks like x can't be negative - this makes sense: you can't have negative variance

# Plotting it (2)

* Adjusting the x-axis

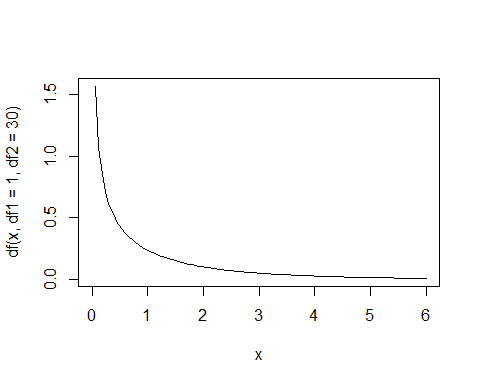
curve(df(x, df1 = 1, df2 = 30), from = 0, to = 6)



# t and F

* Now for something unexpected

curve(df(x, df1 = 1, df2 = 30), from = 0, to = 6)



curve(dt(x, df = 30), from = 0, to = 6)

