

# Analysis of data from a seed bank study in R

The purpose of this small exercise is to create an artificial seed bank dataset and show some principal ways of analysing these data using the **R statistical software** package. Let's assume that the experiment varied the following **factors**:

1. **Scarification Temperature** (Room temperature, 30°C or 50°C)
2. **Duration of Storage** (Control, 10, 20, 40, 80 or 160 days)
3. **Species** (*Plantago boissieri*, *P. ovata*, and *P. ciliata*)

Let us further assume that the experiment is a **fully randomized factorial**, i.e. every factor is randomly allocated to **independent populations** of seed, and **every combination** of factors is allowed. Thus, our experiment comprises

- 3 levels of **temperature**
- 6 levels of **storage**
- 3 levels of **species**
- each factor is replicated 5 times

In **summary**, we have  $n=(3 \times 6 \times 3) \times 5 = 270$  **independent replicates** in our experiment. Thus, there would be 5 petri dishes of seeds stored at room temperature for *P. boissieri*, 5 petri dishes of the same treatment combination, but a different *Plantago* species, and so on. Once we've created our dataset, we can look at a summary table to check if everything is well replicated.

## Setting up the artificial dataframe

Before we can type in (or create) the values of our response variable, we have two options. First, we **could** use a spreadsheet-based program such as **Excel** to create our dataframe. This could look like this:

<b>species</b>	<b>temp</b>	<b>days</b>	<b>germination</b>
<i>P.boissieri</i>	20°	control	
<i>P.boissieri</i>	20°	control	
<i>P.boissieri</i>	20°	control	
<i>P.boissieri</i>	20°	control	
<i>P.boissieri</i>	20°	control	
<i>P.boissieri</i>	20°	10 days	
<i>P.boissieri</i>	20°	10 days	

```
P.boissieri20°      10 days
P.boissieri20°      10 days
P.boissieri20°      10 days
(...)              (...)  (...)
```

The **problem** with such an approach is that it could easily **take us ages** to create that spreadsheet, especially if we were dealing with more than 270 replicates. Thus, **we try our luck using R**, which - as we will soon see - just takes **three lines of code** to create the whole dataframe.

### Creating the dataset in R

We need just a few **basic commands** in R in order to create our dataframe. But unless you don't know exactly what these commands mean, it will be hard for you to grasp what we'll be doing in the next steps. So here comes what you need to know first:

1. The **rep()** command. This says "**repeat** the following X times" - e.g. "print 'P. boissieri' three times. This is how rep() is used:

```
rep("P. boissieri",3)
```

which yields:

```
[1] "P. boissieri" "P. boissieri" "P. boissieri"
```

2. The **c()** command. This says "**concatenate** all elements into one single vector" - or simply "combine". For example, let's say you want to create a series of AAABBBCCC; you could first use **rep("A",3)** and **rep("B",3)** and so on, and then concatenate all three sub-sequences into a single one. An example should make this clear:

```
rep("A",3)
```

```
[1] "A" "A" "A"
```

```
rep("B",3)
```

```
[1] "B" "B" "B"
```

```
rep("C",3)
```

```
[1] "C" "C" "C"
```

Now we use `c()` to combine all three, separating each element by commas:

```
c( rep("A",3), rep("B",3), rep("C",3) )
```

```
[1] "A" "A" "A" "B" "B" "B" "C" "C" "C"
```

3. Finally, we can go one step further and **replicate this last sequence** for an almost infinite time; let's assume we want to have our "AAABBBCCC" sequence replicated **100 times**. This is easy: We just use `rep()` once more, with the syntax "rep(our sequence, 100 times)":

```
rep( c(rep("A",3),rep("B",3),rep("C",3)) ,100)
```

Our output could then look like this:

```
[1] "A" "A" "A" "B" "B" "B" "C" "C" "C" "A" "A" "A" "B" "B" "B" "C" "C" "C" "A"
[20] "A" "A" "B" "B" "B" "C" "C" "C" "A" "A" "A" "B" "B" "B" "C" "C" "C" "A" "A"
[39] "A" "B" "B" "B" "C" "C" "C" "A" "A" "A" "B" "B" "B" "C" "C" "C" "A" "A" "A"
[58] "B" "B" "B" "C" "C" "C" "A" "A" "A" "B" "B" "B" "C" "C" "C" "A" "A" "A" "B"
[77] "B" "B" "C" "C" "C" "A" "A" "A" "B" "B" "B" "C" "C" "C" "A" "A" "A" "B" "B"
[96] "B" "C" "C" "C" "A" "A" "A" "B" "B" "B" "C" "C" "C" "A" "A" "A" "B" "B" "B"
[115] "C" "C" "C" "A" "A" "A" "B" "B" "B" "C" "C" "C" "A" "A" "A" "B" "B" "B" "C" [...]
```

I think you got the point. R enables us to very quickly create huge dataframes and save a lot of time (and money, because it's free).

Knowing all these preliminaries, how can we move on to our specific case with the *Plantago* species? I'll write down the code below, and you can try out on your own what this means:

```
species<-
c(rep("P.boissieri",90),rep("P.ovata",90),rep("P.ciliata",90))

temp<-
rep(c(rep("20°",30),rep("30°",30),rep("50°",30)),3)

days<-
rep(c(rep("control",5),rep("10 days",5),rep("20
days",5),rep("40 days",5),rep("80 days",5),rep("160
days",5)),9)
```

The only thing you might be unfamiliar with is the "gets" operator "<-" which is used to assign names to variables. Thus, the syntax

```
temp<-rep(A,B)
```

means 'replicate A B-times and call this variable "temp"'. That's all you need to know up to here.

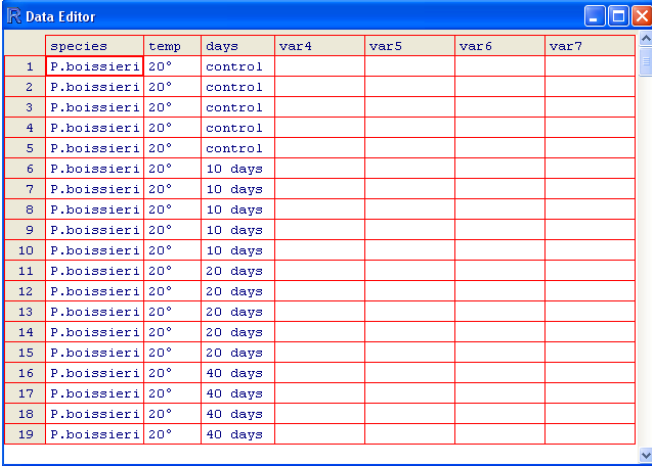
Let's inspect what we have done so far: We have created three variables, **species**, **temp**, and **days**; ideally, we would want to have them combined in a spreadsheet-like manner. Luckily, this is very easy in R: We just type

```
data<-data.frame(species,temp,days)
```

As you see, the "gets" operator, "<-", is used once again here, to give our spreadsheet a name, **data**. To see the result, we use

```
edit(data)
```

which yields the following output:



	species	temp	days	var4	var5	var6	var7
1	P.boissieri	20°	control				
2	P.boissieri	20°	control				
3	P.boissieri	20°	control				
4	P.boissieri	20°	control				
5	P.boissieri	20°	control				
6	P.boissieri	20°	10 days				
7	P.boissieri	20°	10 days				
8	P.boissieri	20°	10 days				
9	P.boissieri	20°	10 days				
10	P.boissieri	20°	10 days				
11	P.boissieri	20°	20 days				
12	P.boissieri	20°	20 days				
13	P.boissieri	20°	20 days				
14	P.boissieri	20°	20 days				
15	P.boissieri	20°	20 days				
16	P.boissieri	20°	40 days				
17	P.boissieri	20°	40 days				
18	P.boissieri	20°	40 days				
19	P.boissieri	20°	40 days				

So everything is fine by now, and we can start typing in data. For the purpose of this exercise, we just quickly **create an artificial variable**, called **germination**, and then move on with the analysis:

```
germination<-sample(seq(0,1,0.01),270,replace=T)*100
```

The sample() command is used to draw 270 random numbers (between 0 and 1 in this case, with steps of 0.01) with replacement. Because all values will lie between 0 and 1, we multiply them by 100 to get pseudo-"percentage" values.

For an overview of what we have so far, we use

```
table(days,species,temp)
```

to see the **number of replicates** in our dataframe called "data" (I will not present the output here, you can try out on your own).

We can also type

```
str(germination)
```

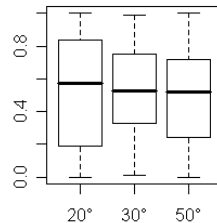
to see how many germination readings we have, and what the values look like:

```
num [1:270] 34 36 64 3 70 82 51 11 91 41 ...
```

Finally, we could try with some first graphical representations of our data, using the `plot()` command. For example, if you type

```
plot(temp,germination)
```

a **graphics window** will open, and you will get the following output:



So everything is fine by now. But now we'd also like to see some **treatment effects**. Let's suppose we want to have a **higher germination rate** for all seeds that have been received a 50°C scarification temperature. We first address all values of the variable "germination" that passes our criterion "50°C", like this:

```
germination[temp=="50°"]
```

We use the **square brackets**, `[],` to address those values of germination that occur at the temperature "50°". *Just for comparison: In Excel, this could for example be achieved using the "Auto Filter" command.*

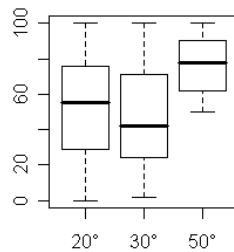
And now we want to sample from a **different range of values**. Let's say, instead of the interval 0...1 we want to have **all values between 0.5 and 1** (i.e. **between 50% and 100% germination**):

```
germination[temp=="50°"]<-sample(seq(0.5,1,0.01),90,replace=T)*100
```

the second line of this command should already be familiar to you; it is the `sample()` command again, with the minor change that we write `seq(0.5,1,0.01),90` instead of `seq(0,1,0.01),270`.

Let's see what has happened:

```
plot(temp,germination)
```



The germination rate for the 50° treatment indeed is now much higher, starting well above the 50% value on the y-axis.

### Analysis of the dataset

Let's recall what **type of experiment** we have: There are **three** so-called **factors** that we varied in order to study their **effects on germination** of seeds:

1. **Scarification Temperature** (3 levels)
2. **Duration of Storage** (6 levels)
3. **Species** (3 levels)

These are the so-called **factors** we manipulated on purpose during the course of our experiment.

Let's ask ourselves: What have we **measured**? Of course, we have "measured" the temperatures, the duration and so on, and we have also "measured" what species we were dealing with. **But this is all uninformative**. We could also have given the species arbitrary names "A", "B", "C". Or we could have said the temperatures were "high", "medium" and "low".

The only thing we are really interested in is the **germination of our seeds**. This is why we call this the "**response variable**" - because "germination" "responded" in some way to our three experimental treatments.

So what consequences does this all have for our analyses? Well, definitely, we're **not** going to do a regression analysis - we have **factors** whose **effects on our response variable** we want to estimate. The **correct** way to analyze these data is a technique called **analysis of variance**. Without going into detail, let's directly start with this kind of analysis, using R.

We have to tell R first that we're dealing with factors. This is easy and involves just three further lines of code:

```
species<-factor(species)
temp<-factor(temp)
```

```
days<-factor(days)
```

The command for an **analysis of variance** is, guess, "aov" (which is just an abbreviation). Now comes the syntax. This is **very, very important** and you should memorize it:

```
Model<-aov(response variable ~ explanatory variable(s))
```

Let's go through this **step by step**.

1. First of all, an **analysis of variance** is a **statistical model** to describe our data. Thus, it is reasonable to give it a name, and because we are lazy, we call it "**Model**".
2. Second comes the **aov()** command
3. In an analysis of variance, we always **type the response variable first**, followed by the "tilda" symbol, "~", which reads "is modelled as a function of...".
4. After the tilda, "~", come the **explanatory variables** (three in our case), separated by a "+". Thus, we model germination as a function of **temp+days+species**:

```
Model<-aov(germination~temp+days+species)
```

Believe it or not, that's all! All we need to do now, to see what our analysis says, is type

```
summary(Model)
```

And we get an output ANOVA table, telling us what **main effects** were **significant**:

```
      Df Sum Sq Mean Sq F value    Pr(>F)
temp    2  40757   20378  30.5518 1.210e-12 ***
days    5    923     185   0.2768   0.9256
species  2   1330     665   0.9970   0.3704
Residuals 260 173423     667
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

## Further steps for more experienced users

There are several issues that I have skipped so far in order to ease understanding.

1. First of all, we could make days an **ordered factor** (we know that "control"<20<40<80<160 days):

```
days<-ordered(days,levels=c("control","10 days","20 days","40 days","80 days","160 days"))
```

This is especially **important for plotting**, because R usually orders the levels of a factor **by alphabet**, unless you tell it the order in which you'd like it to be plotted.

2. Secondly, we would (especially with "real" data) almost certainly have to transform our response variable using an arcsine-square root transformation like this:

```
arcgermination<-asin(sqrt(germination/100))
```

3. Thirdly, we could dive deeper into our data by specifying the anova as a **full factorial ANOVA**, including **interactions**, using "\*" instead of "+" as the separator for the **explanatory variables**:

```
modell<-aov(arcgermination~species*temp*days)
```

4. Further steps would now include model simplification like this:

```
modell<-aov(arcgermination~species*temp*days)
summary(modell)
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
species	2	0.0406	0.0203	0.2106	0.8103
temp	2	5.8526	2.9263	30.3390	2.442e-12 ***
days	5	0.1299	0.0260	0.2694	0.9295
species:temp	4	0.1972	0.0493	0.5110	0.7277
species:days	10	1.0716	0.1072	1.1110	0.3550
temp:days	10	0.2521	0.0252	0.2614	0.9886
species:temp:days	20	1.8622	0.0931	0.9653	0.5056
Residuals	216	20.8339	0.0965		

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
modell2<-update(modell,~.-species:temp:days-temp:days-species:days-species:temp)
```

```
anova(modell2,modell)
```

```
Analysis of Variance Table
```

```
Model 1: arcgermination ~ species + temp + days
```

```
Model 2: arcgermination ~ species * temp * days
```

	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
1	260	24.217				
2	216	20.834	44	3.383	0.7971	0.8135

```
summary(modell2)
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
species	2	0.0406	0.0203	0.2181	0.8042
temp	2	5.8526	2.9263	31.4176	6.012e-13 ***
days	5	0.1299	0.0260	0.2789	0.9244
Residuals	260	24.2169	0.0931		

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
modell3<-update(modell2,~.-species-days)
```

```
anova(modell2,model3)
```



Analysis of Variance Table

```

Model 1: arcgermination ~ species + temp + days
Model 2: arcgermination ~ temp
  Res.Df    RSS Df Sum of Sq    F Pr(>F)
1      260 24.2169
2      267 24.3874 -7   -0.1705 0.2615  0.968

```

```

summary(model3)
              Df Sum Sq Mean Sq F value    Pr(>F)
temp              2   5.8526   2.9263  32.038 3.380e-13 ***
Residuals       267  24.3874   0.0913
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

**Model3** would thus represent the **minimal adequate model**, containing just temp as the only significant explanatory variable.

5. A similar result can be obtained using automated model simplification, i.e. with the **step()** command in R:

```

step(model1)
Start: AIC= -583.7
      arcgermination ~ species * temp * days

              Df Sum of Sq    RSS    AIC
- species:temp:days 20      1.86   22.70 -600.58
<none>                                20.83 -583.70

Step: AIC= -600.58
      arcgermination ~ species + temp + days + species:temp + species:days +
      temp:days

              Df Sum of Sq    RSS    AIC
- temp:days    10      0.25   22.95 -617.60
- species:days 10      1.07   23.77 -608.13
- species:temp   4      0.20   22.89 -606.25
<none>                                22.70 -600.58

Step: AIC= -617.6
      arcgermination ~ species + temp + days + species:temp + species:days

              Df Sum of Sq    RSS    AIC
- species:days 10      1.07   24.02 -625.28
- species:temp   4      0.20   23.15 -623.29
<none>                                22.95 -617.60

Step: AIC= -625.28
      arcgermination ~ species + temp + days + species:temp

              Df Sum of Sq    RSS    AIC
- days          5      0.13   24.15 -633.82
- species:temp   4      0.20   24.22 -631.07
<none>                                24.02 -625.28

Step: AIC= -633.82
      arcgermination ~ species + temp + species:temp

              Df Sum of Sq    RSS    AIC
- species:temp   4      0.20   24.35 -639.63
<none>                                24.15 -633.82

Step: AIC= -639.63
      arcgermination ~ species + temp

              Df Sum of Sq    RSS    AIC
- species        2      0.04   24.39 -643.18
<none>                                24.35 -639.63
- temp           2      5.85   30.20 -585.46

Step: AIC= -643.18

```

```

arcgermination ~ temp

          Df Sum of Sq      RSS      AIC
<none>          24.39 -643.18
- temp      2      5.85  30.24 -589.10
Call:
aov(formula = arcgermination ~ temp)

Terms:
          temp Residuals
Sum of Squares  5.852583 24.387418
Deg. of Freedom      2      267

Residual standard error: 0.3022228
Estimated effects may be unbalanced

```

6. After having established the significant main effect of temp on arcgermination, we can look for significant differences between the levels of temp. This is what is commonly called "multiple comparisons", but there's an easier approach to this, using **summary.lm()** rather than **summary()** to inspect the **aov()** output:

```
summary.lm(model3)
```

---

```

Call:
aov(formula = arcgermination ~ temp)

Residuals:
    Min       1Q   Median       3Q      Max
-0.77310 -0.21414 -0.01902  0.21806  0.79770

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.76232    0.03186  23.929 < 2e-16 ***
temp30°      0.01077    0.04505   0.239  0.811
temp50°      0.31757    0.04505   7.049 1.54e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3022 on 267 degrees of freedom
Multiple R-Squared:  0.1935,    Adjusted R-squared:  0.1875
F-statistic: 32.04 on 2 and 267 DF,  p-value: 3.380e-13

```

---

The most important part of the output are the **coefficients**. The parameter labelled (Intercept) is the mean of arcgermination for the first level of temperature (in our case, 20°C). We can check this by typing

```
mean(arcgermination[temp=="20°"])
[1] 0.762322
```

**Now comes the important bit** (and the part that can cause big headache): The second parameter, labelled **temp30°**, is the **difference** between the second and the first level of temp (30°-20°). This again becomes clear if we calculate it by hand:

```
mean(arcgermination[temp=="30°"])-mean(arcgermination[temp=="20°"])
[1] 0.01077346
```

Likewise, the third parameter is the difference between the means of the 50° and the 20° treatments:

```
mean(arcgermination[temp=="50°"])-
mean(arcgermination[temp=="20°"])
[1] 0.3175662
```

The adjacent columns list the standard errors of these differences. If, for some reason, you want to know the means predicted from the model, you can use

```
model.tables(model3,"means",se=T)
```

```
Grand mean 0.8717685
temp
  20°    30°    50°
0.7623 0.7731 1.0799
```

Or, for the **back-transformed means**, you can directly address these three means, and back-transform them using **sin(y)<sup>2</sup>**:

```
sin(model.tables(model3,"means",se=T)$tables$temp)^2*100
temp
  20°    30°    50°
47.69320 48.76985 77.77565
```

7. Finally, if you really want to do more, you can do **multiple comparisons**, yielding the same results, using one or all of the following commands:

```
plot(TukeyHSD(model3))
TukeyHSD(model3)

pairwise.t.test(arcgermination,temp,
adjust.method="Bonferroni")
```

In case you're unfamiliar with these commands, use `?TukeyHSD` or `?pairwise.t.test` for more information.

**Good luck!**