

Troubleshooting (in R) Seminar

Josh Sumner

Outline

- **Patrick's Lollipop/Occam's Razor**
- **General Troubleshooting steps**
- **Troubleshooting in R**
 - Installations
 - Reading in data/Reshaping data
 - Data types
 - Booleans and Assignment
 - ggplot2
 - Loops/apply
 - Basic stats
 - Model making

You troubleshoot all the time.

- If your coffee isn't right in the morning you:
 - Check if there is water in the pot
 - Check if it's plugged in
 - Check if the coffee is ground
 - Check if the coffee is in the basket
 - Check if there is a filter
 - Etc, you troubleshoot based on a problem you identified and start with easiest things to check.
 - Your first step isn't to rewire the heating element.

So there you were...

- You're on the run from the long arm of the law

So there you were...

- You're on the run from the long arm of the law
- You have very limited supplies to share with your partner in crime

So there you were...

- You're on the run from the long arm of the law
- You have very limited supplies to share with your partner in crime
- But your food is missing...

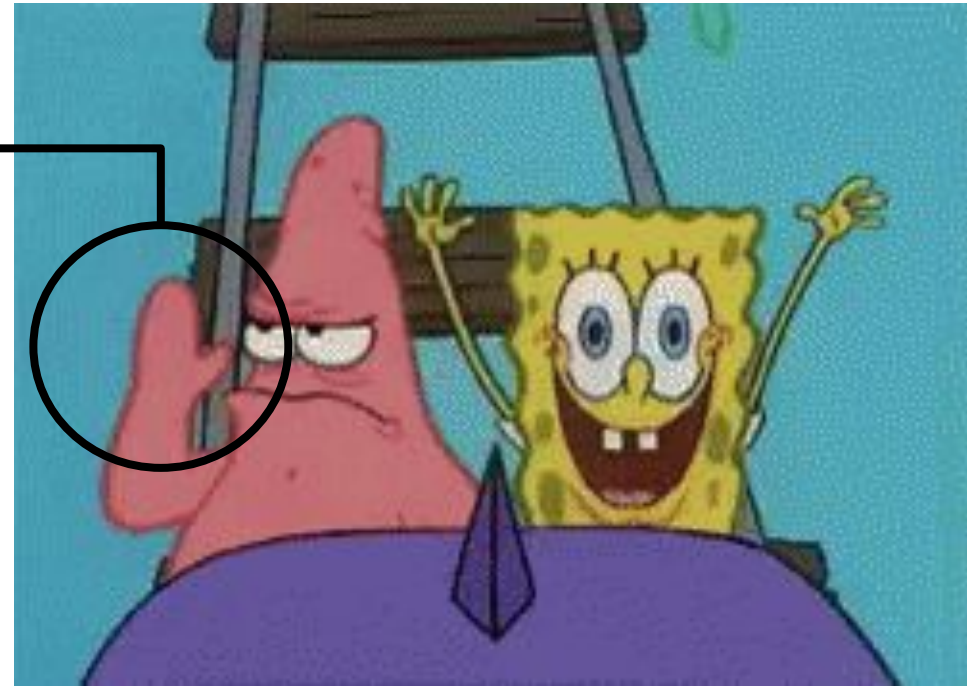
- Let's troubleshoot before someone gets hurt



- You definitely had a bar of chocolate a minute ago.



- You definitely had a bar of chocolate a minute ago.
- Now you don't have anything. Something has to have happened in the last few minutes.



- You definitely had a bar of chocolate a minute ago.
 - In here several things could have happened:
 - It was stolen
 - You lost it
 - You ate it
 - You're misremembering
 - Other?
- Now you don't have anything. Something has to have happened in the last few minutes.



Something went wrong between points A and B



Had food



No food



Something went wrong between points A and B



Had food



No food

Maybe we are
wrong a-priori?

Something went wrong between points A and B



Had food



No food

Maybe we are wrong a-priori?

Something went wrong between points A and B



Had food

No food

Maybe we are
wrong a priori

Maybe we
dropped it?

Something went wrong between points A and B



Had food

No food

Maybe we are wrong a priori

Maybe we dropped it?



Something went wrong between points A and B



Had food

No food

Maybe we are wrong a priori

Maybe we dropped it?

Was it stolen?



Something went wrong between points A and B



Had food

No food

Maybe we are wrong a priori

Maybe we dropped it?

Was it stolen?

Something went wrong between points A and B



Had food

No food

Maybe we are
wrong a priori

Maybe we
dropped it?

Was it stolen?

Something went wrong between points A and B



Had food

No food

Maybe we are
wrong a priori

Maybe we
dropped it?

Was it stolen?

Did we eat it?

Something went wrong between points A and B



Had food

No food

Maybe we are wrong a priori

Maybe we dropped it?

Was it stolen?

Did we eat it?

Something went wrong between points A and B



Had food

No food

Maybe we are wrong a priori

Maybe we dropped it?

Was it stolen?

Did we eat it?

Next time we'll know what happened



Did we eat it?

"Fool me once shame on... you, fool me - you can't get fooled again" - GWB

Applying your skill to code

- You understand troubleshooting in day to day life, but how does it apply to code?

Applying your skill to code

- You understand troubleshooting in day to day life, but how does it apply to code?
- We will try to identify steps and guidelines to use when problems come up.

Outline

- Patrick's Lollipop/Occam's Razor
- **General Troubleshooting steps**
- **Troubleshooting in R**
 - Installations
 - Reading in data/Reshaping data
 - Data types
 - Booleans and Assignment
 - ggplot2
 - Loops/apply
 - Basic stats
 - Model making

General Troubleshooting Steps

- Read the error
- Find the problem area
- Simplify/Get verbose
- Make edits
- Batch tests

General Troubleshooting Steps

We have some area values in pixels and want to classify plants as germinated or ungerminated

```
> summary(area_in_px)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.00   0.00   16.50   19.83   33.25   92.00
>
> germination <- if(area_in_px > 10) { "Ungerminated" else {"Germinated" } }
```

General Troubleshooting Steps

We have some area values in pixels and want to classify plants as germinated or ungerminated, but we get this error:

```
> summary(area_in_px)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.00   0.00   16.50   19.83   33.25   92.00
>
> germination <- if(area_in_px > 10) { "Ungerminated" else {"Germinated" }
Error: unexpected 'else' in "germination <- if(area_in_px > 10) { "Ungerminated" else"
```

- **Read Error**
- **Find Area**
- **Simplify**
- **Edit**
- **Test**

General Troubleshooting Steps

We have some area values in pixels and want to classify plants as germinated or ungerminated, but we get this error:

```
> summary(area_in_px)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.00   0.00   16.50   19.83   33.25   92.00
>
> germination <- if(area_in_px > 10) { "Ungerminated" else {"Germinated" }
Error: unexpected 'else' in "germination <- if(area_in_px > 10) { "Ungerminated" else"
> germination <- if(area_in_px > 10) { "Ungerminated"} else {"Germinated" }
```

- Read Error
- Find Area
- Simplify
- Edit
- Test

General Troubleshooting Steps

We have some area values in pixels and want to classify plants as germinated or ungerminated, but we get this error:

```
> summary(area_in_px)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.00   0.00   16.50   19.83   33.25   92.00

>
> germination <- if(area_in_px > 10) { "Ungerminated" else {"Germinated" }
Error: unexpected 'else' in "germination <- if(area_in_px > 10) { "Ungerminated" else"

> germination <- if(area_in_px > 10) { "Ungerminated"} else {"Germinated" }
Error in if (area_in_px > 10) { : the condition has length > 1
```

- Read Error
- Find Area
- Simplify
- Edit
- Test

General Troubleshooting Steps

We have some area values in pixels and want to classify plants as germinated or ungerminated, but we get this error:

```
> summary(area_in_px)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.00   0.00   16.50   19.83   33.25   92.00
>
> germination <- if(area_in_px > 10) { "Ungerminated" else {"Germinated" }
Error: unexpected 'else' in "germination <- if(area_in_px > 10) { "Ungerminated" else"
> germination <- if(area_in_px > 10) { "Ungerminated"} else {"Germinated" }
Error in if (area_in_px > 10) { : the condition has length > 1
```

- Read Error
- Find Area
- Simplify
- Edit
- Test

General Troubleshooting Steps

```
> germination <- if(area_in_px > 10) { "Ungerminated"} else {"Germinated" }  
Error in if (area_in_px > 10) { : the condition has length > 1
```

```
> ?`if`  
> |
```



Usage

```
if(cond) expr  
if(cond) cons.expr else alt.expr
```

```
for(var in seq) expr  
while(cond) expr  
repeat expr  
break  
next
```

Arguments

cond

A length-one logical vector that is not NA. Other types are coerced to logical if possible, ignoring any class. (Conditions of length greater than one are an error.)

- Read Error
- Find Area
- Simplify
- Edit
- Test

General Troubleshooting Steps

```
> germination <- if(area_in_px > 10) { "Ungerminated"} else {"Germinated" }  
Error in if (area_in_px > 10) { : the condition has length > 1
```

See Also

[Syntax](#) for the basic **R** syntax and operators, [Paren](#) for parentheses and braces.

[ifelse](#), [switch](#) for other ways to control flow.

- Read Error
- Find Area
- Simplify
- Edit
- Test

General Troubleshooting Steps

```
> germination <- ifelse(area_in_px > 10, "Ungerminated", "Germinated")
```

- Read Error
- Find Area
- Simplify
- [Edit](#)
- Test

General Troubleshooting Steps

```
> germination <- ifelse(area_in_px > 10, "Ungerminated", "Germinated")
> table(germination)
germination
  Germinated Ungerminated
           37           63
```

- Read Error
- Find Area
- Simplify
- Edit
- Test

General Troubleshooting Steps

Troubleshooting is easier when there is documentation to check.

In R it's common to write your own functions, either in apply functions or to use later. Now we'll go through a quick example using one of those.

In the next example `inner1` and `inner2` are used to simulate data.

General Troubleshooting Steps

```
inner1 <- function(x,a,b,c){  
  a_r <- a+rnorm(1,mean = 0,sd=10)  
  b_r <- b+rnorm(1,mean=0,sd=2)  
  c_r <- c+rnorm(1,mean=0,sd=.035)  
  return(a_r*exp(-b_r*exp(-c_r*x))) }  
inner2 <- function(i, geno, trt, times, a, b,c,saf){  
  data.frame("sample"=paste0("sample_",i),"genotype"=geno, treatment=trt,  
            "time"=times,"y"=inner1(times,a,b,c),stringsAsFactors = "") }  
df <- do.call(rbind, lapply(1:20, function(i){ inner2(i, "genoA", "trtA", 1:31, 200, 14, 0.2) })))
```

If you are using Rstudio then some errors will be flagged automatically and Traceback is available in the console



General Troubleshooting Steps

```
inner1 <- function(x,a,b,c){  
  a_r <- a+rnorm(1,mean = 0,sd=10)  
  b_r <- b+rnorm(1,mean=0,sd=2)  
  c_r <- c+rnorm(1,mean=0,sd=.035)  
  return(a_r*exp(-b_r*exp(-c_r*x))) }  
inner2 <- function(i, geno, trt, times, a, b,c,saf){  
  data.frame("sample"=paste0("sample_",i),"genotype"=geno, treatment=trt,  
            "time"=times,"y"=inner1(times,a,b,c),stringsAsFactors = "") }  
df <- do.call(rbind, lapply(1:20, function(i){ inner2(i, "genoA", "trtA", 1:31, 200, 14, 0.2) })))
```

If you are using Rstudio then some errors will be flagged automatically and Traceback is available in the console

```
> df <- do.call(rbind, lapply(1:20, function(i){ inner2(i, "genoA", "trtA", 1:31, 200,  
14, 0.2) })))
```



```
Error in if (stringsAsFactors) x <- factor(x) :  
argument is not interpretable as logical
```

 Show Traceback
 Rerun with Debug

General Troubleshooting Steps

```
> df <- do.call(rbind, lapply(1:20, function(i){ inner2(i, "genoA", "trtA", 1:31, 200,
14, 0.2) })))
```

Error in if (stringsAsFactors) x <- factor(x) :
argument is not interpretable as logical

 Show Traceback
 Rerun with Debug



```
7. as.data.frame.character(x[[i]], optional = TRUE, stringsAsFactors = stringsAsFactors)
6. as.data.frame(x[[i]], optional = TRUE, stringsAsFactors = stringsAsFactors)
5. data.frame(sample = paste0("sample_", i), genotype = geno, treatment = trt,
  time = times, y = inner1(times, a, b, c), stringsAsFactors = "")
4. inner2(i, "genoA", "trtA", 1:31, 200, 14, 0.2)
3. FUN(X[[i]], ...)
2. lapply(1:20, function(i) {
  inner2(i, "genoA", "trtA", 1:31, 200, 14, 0.2)
})
1. do.call(rbind, lapply(1:20, function(i) {
  inner2(i, "genoA", "trtA", 1:31, 200, 14, 0.2)
})))
```



General Troubleshooting Steps

```
> df <- do.call(rbind, lapply(1:20, function(i){ inner2(i, "genoA", "trtA", 1:31, 200,
14, 0.2) })))
```

```
Error in if (stringsAsFactors) x <- factor(x) :  
argument is not interpretable as logical
```

 Show Traceback
 Rerun with Debug

```
7. as.data.frame.character(x[[i]], optional = TRUE, stringsAsFactors = stringsAsFa  
ctors)  
6. as.data.frame(x[[i]], optional = TRUE, stringsAsFactors = stringsAsFactors)  
5. data.frame(sample = paste0("sample_", i), genotype = geno, treatment = trt,  
time = times, y = inner1(times, a, b, c), stringsAsFactors = "")  
4. inner2(i, "genoA", "trtA", 1:31, 200, 14, 0.2)  
3. FUN(X[[i]], ...)  
2. lapply(1:20, function(i) {  
  inner2(i, "genoA", "trtA", 1:31, 200, 14, 0.2)  
})  
1. do.call(rbind, lapply(1:20, function(i) {  
  inner2(i, "genoA", "trtA", 1:31, 200, 14, 0.2)  
})))
```

Call to a base
R function

Called function



do.call(rbind
was fine

Original Call

General Troubleshooting Steps

```
> df <- do.call(rbind, lapply(1:20, function(i){ inner2(i, "genoA", "trtA", 1:31, 200,
14, 0.2) })))
```

```
Error in if (stringsAsFactors) x <- factor(x) :  
argument is not interpretable as logical
```

 Show Traceback
 Rerun with Debug

```
7. as.data.frame.character(x[[i]], optional = TRUE, stringsAsFactors = stringsAsFactors)
6. as.data.frame(x[[i]], optional = TRUE, stringsAsFactors = stringsAsFactors)
5. data.frame(sample = paste0("sample_", i), genotype = geno, treatment = trt,
  time = times, y = inner1(times, a, b, c), stringsAsFactors = "")
4. inner2(i, "genoA", "trtA", 1:31, 200, 14, 0.2)
3. FUN(X[[i]], ...)
2. lapply(1:20, function(i) {
  inner2(i, "genoA", "trtA", 1:31, 200, 14, 0.2)
})
1. do.call(rbind, lapply(1:20, function(i) {
  inner2(i, "genoA", "trtA", 1:31, 200, 14, 0.2)
})))
```

We know base R works, so
If R got that far it means the
arguments given to that
Base R function are the problem

We see that it tried to make a
Logical comparison to "" so
we fix that argument in inner2.

General Troubleshooting Steps

```
> mean(numbers)
[1] NA
Warning message:
In mean.default(numbers) : argument is not numeric or logical: returning NA
> str(numbers)
chr [1:1000] "T" "Y" "P" "X" "V" "K" "P" "T" "H" "C" "D" "T" "L" "V" "Q" "J" "T" "K" "Y" "W" ...
> head(numbers)
[1] "T" "Y" "P" "X" "V" "K"
> class(numbers)
[1] "character"
```

One of the most common Problems to run into is just That the object you're using is Not the type of object you Wanted to use.

Str(), class(), head(), etc are All good ways to check that.

Outline

- Patrick's Lollipop/Occam's Razor
- General Troubleshooting steps
- **Troubleshooting in R**
 - Installations
 - Reading in data/Reshaping data
 - Data types
 - Booleans and Assignment
 - ggplot2
 - Loops/apply
 - Basic stats
 - Model making

But first, take a break. Come back in some amount of minutes.

Scenario format

- Today we'll go through several exercises following a scenario where you want to read in several files of related image analysis data which we'll then plot, reshape, and analyze.
- For each scenario work with your group to solve the problem(s) then we'll go over solutions together.

Scenario 1

- Several members of your lab have been running germination experiments and you are going to collate the data and compare treatments.
 - What could go wrong?

Scenario 1: Exercise 1

- We know we'll need to make some ggplot figures so we start installing packages, troubleshoot this installation.

```
> library(ggplot)
Error in library(ggplot) : there is no package called 'ggplot'
> install.packages("ggplot")
Installing package into '/home/josh/R/x86_64-pc-linux-gnu-library/4.3'
(as 'lib' is unspecified)
Warning in install.packages :
  package 'ggplot' is not available for this version of R
```

A version of this package for your version of R might be available elsewhere,
see the ideas at
<https://cran.r-project.org/doc/manuals/r-patched/R-admin.html#Installing-packages>

Scenario 1: Exercise 1

- We know we'll need to make some ggplot figures so we start installing packages, troubleshoot this installation.
 - What are your first steps?

```
> library(ggplot)
Error in library(ggplot) : there is no package called 'ggplot'
> install.packages("ggplot")
Installing package into '/home/josh/R/x86_64-pc-linux-gnu-library/4.3'
(as 'lib' is unspecified)
Warning in install.packages :
  package 'ggplot' is not available for this version of R
```

A version of this package for your version of R might be available elsewhere, see the ideas at <https://cran.r-project.org/doc/manuals/r-patched/R-admin.html#Installing-packages>

Scenario 1: Exercise 1

- We know we'll need to make some ggplot figures so we start installing packages, troubleshoot this installation.
 - What are your first steps?
 - We check CRAN for “ggplot” to see if we made a spelling mistake

```
> pkgs<-names(available.packages()[,"Package"])
> pkgs[grepl("ggplot", pkgs)]
```

[1] "add2ggplot"	"ggplot.multistats"	"ggplot2"
[4] "ggplot2.utils"	"ggplot2movies"	"ggplotAssist"
[7] "ggplotgui"	"ggplotify"	"ggplotlyExtra"
[10] "loon.ggplot"	"qtl2ggplot"	"RcmdrPlugin.KMggplot2"

[Solution Next](#)

Scenario 1: Exercise 1

- Report back on your group's answer

Solution Next

Scenario 1: Exercise 1 - Solution

- We know we'll need to make some ggplot figures so we start installing packages, troubleshoot this installation.
 - What are your first steps?
 - We check CRAN for “ggplot” to see if we made a spelling mistake

```
> install.packages("ggplot2")
```

```
Installing package into '/home/josh/R/x86_64-pc-linux-gnu-library/4.3'  
(as 'lib' is unspecified)
```

```
The downloaded source packages are in  
      '/tmp/Rtmpw10N2D/downloaded_packages'
```

```
> library(ggplot2)
```

Scenario 1: Exercise 1

- We know we'll need to make some ggplot figures so we start installing packages, troubleshoot this installation.
 - What are your first steps?
 - We check CRAN for “ggplot” to see if we made a spelling mistake
 - Go ahead and grab tidyr and data.table while we're at it.

Scenario 1: Extra 1

- While you're at it you figure you'll grab a few other packages you've been meaning to use.
 - Install `Biostrings`

```
> install.packages("Biostrings")
Installing package into '/home/josh/R/x86_64-pc-linux-gnu-library/4.3'
(as 'lib' is unspecified)
Warning in install.packages :
  package 'Biostrings' is not available for this version of R
```

A version of this package for your version of R might be available elsewhere,
see the ideas at
<https://cran.r-project.org/doc/manuals/r-patched/R-admin.html#Installing-packages>

Scenario 1: Extra 1

- While you're at it you figure you'll grab a few other packages you've been meaning to use.
 - Install `Biostrings`
 - Google tells you this comes from Bioconductor

```
> install.packages("Biostrings")  
Installing package into '/home/josh/R/x86_64-pc-linux-gnu-library/4.3'  
(as 'lib' is unspecified)  
Warning in install.packages :  
  package 'Biostrings' is not available for this version of R
```

Solution Next

A version of this package for your version of R might be available elsewhere,
see the ideas at
<https://cran.r-project.org/doc/manuals/r-patched/R-admin.html#Installing-packages>

Scenario 1: Extra 1

- Report back on your group's answer

Solution Next

Scenario 1: Extra 1 - Solution

- While you're at it you figure you'll grab a few other packages you've been meaning to use.
 - Install `Biostrings`
 - Google tells you this comes from Bioconductor

```
> BiocManager::available("Biostrings")
'getOption("repos")' replaces Bioconductor standard repositories, see 'help("repositories",
package = "BiocManager")' for details.
Replacement repositories:
  CRAN: https://cloud.r-project.org
[1] "Biostrings"

> BiocManager::install("Biostrings")
'getOption("repos")' replaces Bioconductor standard repositories, see 'help("repositories",
package = "BiocManager")' for details.
Replacement repositories:
  CRAN: https://cloud.r-project.org
Bioconductor version 3.18 (BiocManager 1.30.22), R 4.3.2 (2023-10-31)
```

Scenario 1: Extra 2

- While you're at it you figure you'll grab a few other packages you've been meaning to use.
 - Install ``COZINE``

I picked this because it lives in dependency hell. You would need to look at the source code to install this.

Scenario 1: Extra 2

- While you're at it you figure you'll grab a few other packages you've been meaning to use.
 - Install `COZINE`

```
> library(COZINE)
Error in library(COZINE) : there is no package called 'COZINE'
> install.packages("COZINE")
Installing package into '/home/josh/R/x86_64-pc-linux-gnu-library/4.3'
(as 'lib' is unspecified)
Warning in install.packages :
  package 'COZINE' is not available for this version of R
```

A version of this package for your version of R might be available elsewhere, see the ideas at <https://cran.r-project.org/doc/manuals/r-patched/R-admin.html#Installing-packages>

Scenario 1: Extra 2

- While you're at it you figure you'll grab a few other packages you've been meaning to use.
 - Install `COZINE`

```
> BiocManager::available("COZINE")  
'getOption("repos")' replaces Bioconductor standard repositories, see 'help("repos")'  
package = "BiocManager")' for details.  
Replacement repositories:  
  CRAN: https://cloud.r-project.org  
character(0)
```

Scenario 1: Extra 2

- While you're at it you figure you'll grab a few other packages you've been meaning to use.
 - Install `COZINE`

```
> library(devtools)
Loading required package: usethis
> devtools::install_github("MinJinHa/COZINE")
Downloading GitHub repo MinJinHa/COZINE@HEAD
```

Scenario 1: Extra 2

- While you're at it you figure you'll grab a few other packages you've been meaning to use.
 - Install `COZINE`

```
> library(devtools)
Loading required package: usethis
> devtools::install_github("MinJinHa/COZINE")
Downloading GitHub repo MinJinHa/COZINE@HEAD
```

```
Installing package into '/home/josh/R/x86_64-pc-linux-gnu-library/4.3'
(as 'lib' is unspecified)
ERROR: dependency 'HurdleNormal' is not available for package 'COZINE'
* removing '/home/josh/R/x86_64-pc-linux-gnu-library/4.3/COZINE'
```

Scenario 1: Extra 2

- While you're at it you figure you'll grab a few other packages you've been meaning to use.
 - Install `COZINE`

```
> BiocManager::available("HurdleNormal")
'getOption("repos")' replaces Bioconductor standard repositories, see 'help("repos"
package = "BiocManager")' for details.
Replacement repositories:
  CRAN: https://cloud.r-project.org
character(0)
> pkgs<-names(available.packages()[,"Package"])
> pkgs[grepl("HurdleNormal", pkgs)]
character(0)
```

Solution Next

Scenario 1: Extra 2 - Solution

- While you're at it you figure you'll grab a few other packages you've been meaning to use.
 - Install `COZINE`

```
> devtools::install_github('amcdavid/HurdleNormal')  
Downloading GitHub repo amcdavid/HurdleNormal@HEAD
```

```
> devtools::install_github("MinJinHa/COZINE")  
Downloading GitHub repo MinJinHa/COZINE@HEAD
```

```
** testing if installed package keeps a record of temporary installation path  
* DONE (COZINE)
```

Scenario 1: Exercise 2

- The data you need was collected in 4 experiments by 4 different people, you need to read it in and make a single dataframe for visualization/analyses.
 - You should end up with a 960 x 5 dataframe

```
> dim(e2)
```

```
[1] 972  6
```

```
> head(e2)
```

	experiment	trt	DAP	value	variable	NA
1	0	y	1	1.0000000	germ_pct	<NA>
2	0	y	1	1.0000000	hazard_pct	<NA>
3	0	y	1	1.0000000	mean_area	<NA>

```
> dir("data", pattern = "scenario1_*")
```

```
[1] "scenario1_exp1.csv" "scenario1_exp2.txt" "scenario1_exp3.txt" "scenario1_exp4.csv"
```

```
>
```

```
> files <- dir("data", pattern = "scenario1_*", full.names = TRUE)
```

```
> e2 <- do.call(rbind, lapply(files, read.delim))
```

```
Error in match.names(clabs, names(xi)) :
```

```
names do not match previous names
```

Simplify and make verbose

Scenario 1: Exercise 2

- The data you need was collected in 4 experiments by 4 different people, you need to read it in and make a single dataframe for visualization/analyses.

```
> for(file in files){
+   print(head(read.delim(file),3))
+ }
experiment.trt.DAP.value.variable
1          1,y,1,0,germ_pct
2          1,y,1,1,hazard_pct
3 1,y,1,0.112095129310443,mean_area
Experiment.treatment.DAP.Value.Variable
1          241&2&y&1&0&germ_pct
2          242&2&y&1&1&hazard_pct
3 243&2&y&1&0.190281908638225&mean_area
expPASSWORDtrtPASSWORDDAPPASSWORDvaluePASSWORDvariable
1          481PASSWORD3PASSWORDyPASSWORD1PASSWORD0PASSWORDgerm_pct
2          482PASSWORD3PASSWORDyPASSWORD1PASSWORD1PASSWORDhazard_pct
3 483PASSWORD3PASSWORDyPASSWORD1PASSWORD0.238540514591839PASSWORDmean_area
X.EXPERIMENT.TREATMENT.DAP.VAL.VAR
1          721,4,y,1,0,germ_pct
2          722,4,y,1,1,hazard_pct
3 723,4,y,1,0.176391548106468,mean_area
```

Edit and Test

Scenario 1: Exercise 2

- The data you need was collected in 4 experiments by 4 different people, you need to read it in and make a single dataframe for visualization/analyses.

```
exp1<-read.csv("data/scenario1_exp1.csv")  
exp2<-read.table("data/scenario1_exp2.txt")  
exp3<-read.table("data/scenario1_exp3.txt")  
exp4<-read.csv("data/scenario1_exp4.csv")
```

```
e2 <- rbind(exp1, exp2, exp3, exp4)
```

```
Error in rbind(deparse.level, ...) :  
  numbers of columns of arguments do not match
```

[Solution Next](#)

Scenario 1: Exercise 2

- Report back on your group's answer/progress

Solution Next

Scenario 1: Exercise 2 - Solution

- The data you need was collected in 4 experiments by 4 different people, you need to read it in and make a single dataframe for visualization/analyses.

```
exp1<-read.csv("data/scenario1_exp1.csv")  
exp2<-read.table("data/scenario1_exp2.txt", sep="&")
```

```
> for(file in files){  
+   print(head(read.delim(file),3))  
+ }  
  experiment.trt.DAP.value.variable  
1           1,y,1,0,germ_pct  
2           1,y,1,1,hazard_pct  
3 1,y,1,0.112095129310443,mean_area  
  Experiment.treatment.DAP.Value.Variable  
1           241&2&y&1&0&germ_pct  
2           242&2&y&1&1&hazard_pct  
3    243&2&y&1&0.190281908638225&mean area
```

We specify the field separator as & after looking at the beginning of the file.

Scenario 1: Exercise 2 - Solution

- The data you need was collected in 4 experiments by 4 different people, you need to read it in and make a single dataframe for visualization/analyses.

```
exp1<-read.csv("data/scenario1_exp1.csv")
exp2<-read.table("data/scenario1_exp2.txt", sep("&"))
lines <- readLines("data/scenario1_exp3.txt")
txt <- gsub("PASSWORD", " ", lines)
dat <- textConnection(txt)
exp3 <- read.table(dat)
```

Here we also have to specify the field separator, but it's multiple character which is extremely stupid, so we need to work Around that.

Here we use `gsub` to replace PASSWORD then read the text as a table.

```
expPASSWORDtrtpASSWORDDDAPPASSWORDvaluePASSWORDvariable
1 481PASSWORD3PASSWORDyPASSWORD1PASSWORD0PASSWORDgerm_pct
2 482PASSWORD3PASSWORDyPASSWORD1PASSWORD1PASSWORDhazard_pct
3 483PASSWORD3PASSWORDyPASSWORD1PASSWORD0.238540514591839PASSWORDmean_area
```

Scenario 1: Exercise 2 - Solution

- The data you need was collected in 4 experiments by 4 different people, you need to read it in and make a single dataframe for visualization/analyses.

```
exp1<-read.csv("data/scenario1_exp1.csv")
exp2<-read.table("data/scenario1_exp2.txt", sep="&")
lines <- readLines("data/scenario1_exp3.txt")
txt <- gsub("PASSWORD", " ", lines)
dat <- textConnection(txt)
exp3 <- read.table(dat)
exp4<-read.csv("data/scenario1_exp4.csv", row.names=1)
```

This one just has an "X" column at the first position. This is very common when data are written out from R using `write.csv` with `row.names=TRUE`, which is the default.

It's easy to work around with `read.csv`.

Scenario 1: Exercise 2 - Solution

- The data you need was collected in 4 experiments by 4 different people, you need to read it in and make a single dataframe for visualization/analyses.

```
exp1<-read.csv("data/scenario1_exp1.csv")
exp2<-read.table("data/scenario1_exp2.txt", sep("&")
lines <- readLines("data/scenario1_exp3.txt")
txt <- gsub("PASSWORD", " ", lines)
dat <- textConnection(txt)
exp3 <- read.table(dat)
exp4<-read.csv("data/scenario1_exp4.csv", row.names=1)

colnames(exp1)<-colnames(exp2)<-colnames(exp3)<-colnames(exp4)<-c("experiment", "trt", "DAP",
                                                                "value", "variable", "type")

e2<-rbind(exp1, exp2, exp3, exp4)

dim(e2)
[1] 972  6
```

We have the data, but we need to trim it down some.

Scenario 1: Exercise 2.5

- Subset your data for only cases where type is “plant” using `tidyr`, `data.table`, and base R.

```
> head(e2)
  experiment trt DAP      value  variable  type
1          0  y   1 1.00000000  germ_pct <NA>
2          0  y   1 1.00000000 hazard_pct <NA>
3          0  y   1 1.00000000 mean_area <NA>
4          1  y   1 0.00000000  germ_pct plant
5          1  y   1 1.00000000 hazard_pct plant
6          1  y   1 0.1120951  mean_area plant
>
```

Scenario 1: Exercise 2.5

- Subset your data for only cases where type is “plant” using `tidyr`, `data.table`, and base R.

```
> head(e2)
  experiment trt DAP      value variable type
1          0  y   1 1.00000000  germ_pct <NA>
2          0  y   1 1.00000000 hazard_pct <NA>
3          0  y   1 1.00000000 mean_area <NA>
4          1  y   1 0.00000000  germ_pct plant
5          1  y   1 1.00000000 hazard_pct plant
6          1  y   1 0.1120951  mean_area plant
>
```

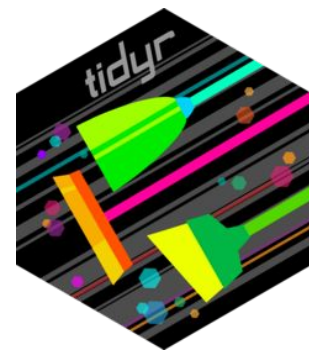
- We'll assign an approach to each group, these will change through the workshop so if you're uncomfortable with your current option the next example will be different.

Scenario 1: Exercise 2.5

- Subset your data for only cases where type is “plant” using `tidyr`, `data.table`, and base R.

```
> head(e2)
  experiment trt DAP      value  variable  type
1          0   y   1 1.0000000    germ_pct <NA>
2          0   y   1 1.0000000   hazard_pct <NA>
3          0   y   1 1.0000000   mean_area <NA>
4          1   y   1 0.0000000    germ_pct plant
5          1   y   1 1.0000000   hazard_pct plant
6          1   y   1 0.1120951   mean_area plant
>
```

- We'll assign an approach to each group, these will change through the workshop so if you're uncomfortable with your current option the next example will be different.
- In your group check the `examples.R` file and start on your version of this problem.



Scenario 1: Exercise 2.5

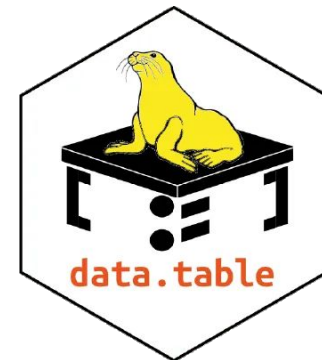
- Subset your data for only cases where type is “plant” using **tidyr**, **data.table**, and base R.

```
> library(dplyr)
> t2.5 <- e2 %>%
+   select(type == "plant")
Error in `select()`:
! Problem while evaluating `type == "plant"`.
Caused by error:
! object 'type' not found
Run `rlang::last_trace()` to see where the error occurred.
```

Scenario 1: Exercise 2.5 - Solution

- Subset your data for only cases where type is “plant” using `tidyr`, `data.table`, and base R.

```
> library(dplyr)
> t2.5 <- e2 %>%
+   filter(type == "plant")
> dim(t2.5)
[1] 960    6
> unique(t2.5$type)
[1] "plant"
```



Scenario 1: Exercise 2.5

- Subset your data for only cases where type is “plant” using `tidyr`, `data.table`, and base R.

```
> library(data.table)
> d2 <- as.data.table(e2)
> d2.5 <- d2[type = "plant",]
Error in `[.data.table`](d2, type = "plant", ) :
  unused argument (type = "plant")
```

Scenario 1: Exercise 2.5 - Solution

- Subset your data for only cases where type is “plant” using `tidyr`, `data.table`, and base R.

```
> library(data.table)
> d2 <- as.data.table(e2)
> d2.5 <- d2[type == "plant",]
> dim(d2.5)
[1] 960    6
> unique(d2.5$type)
[1] "plant"
```



Scenario 1: Exercise 2.5

- Subset your data for only cases where type is “plant” using `tidyr`, `data.table`, and [base R](#).

```
> e2.5 <- e2[e2$type == "plant",]  
> dim(e2.5)  
[1] 972    6  
> unique(e2.5$type)  
[1] NA      "plant"
```



Scenario 1: Exercise 2.5

- Subset your data for only cases where type is “plant” using `tidyr`, `data.table`, and [base R](#).

```
> e2.5 <- e2[e2$type == "plant",]
> dim(e2.5)
[1] 972    6
> unique(e2.5$type)
[1] NA      "plant"
> head(e2.5)
```

	experiment	trt	DAP	value	variable	type
NA	NA	<NA>	NA	NA	<NA>	<NA>
NA.1	NA	<NA>	NA	NA	<NA>	<NA>
NA.2	NA	<NA>	NA	NA	<NA>	<NA>
4	1	y	1	0.0000000	germ_pct	plant
5	1	y	1	1.0000000	hazard_pct	plant
6	1	y	1	0.1120951	mean_area	plant



Scenario 1: Exercise 2.5

- Subset your data for only cases where type is “plant” using `tidyr`, `data.table`, and [base R](#).

```
> 2==2
[1] TRUE
> NA == NA
[1] NA
> is.na(NA)
[1] TRUE
> is.na(NULL)
logical(0)
> NULL==NULL
logical(0)
> is.null(NULL)
[1] TRUE
```

- NULL and NA are special values in R. Tibbles ([tidyverse](#)) and `data.tables` ([data.table](#)) both handle logical comparisons differently than base R for these cases. Here we cannot use ``==``, instead we use ``is.na()`` but ``which()`` could also work.

Solution Next

Scenario 1: Exercise 2.5 - Solution

- Subset your data for only cases where type is “plant” using `tidyr`, `data.table`, and [base R](#).

```
> e2.5 <- e2[!is.na(e2$type),]
> dim(e2.5)
[1] 960    6
> head(e2.5)
```

	experiment	trt	DAP	value	variable	type
4	1	y	1	0.00000000	germ_pct	plant
5	1	y	1	1.00000000	hazard_pct	plant
6	1	y	1	0.1120951	mean_area	plant
7	1	y	2	0.1200000	germ_pct	plant
8	1	y	2	0.8800000	hazard_pct	plant
9	1	y	2	6.0952852	mean_area	plant

- Now you can identify and address one of the two most annoying R errors.
 - That's called foreshadowing

Scenario 1: Exercise 3

- Great, you have the data! But it's in long format and you want it in wide. Widen the data using `tidyr`, `data.table`, and base R.
 - You should end up with a 320 x 6 data.frame

```
> head(e3)
  experiment trt DAP germ_pct hazard_pct mean_area
1          1  y   1    0.00         1.00  0.1120951
2          1  y   2    0.12         0.88  6.0952852
3          1  y   3    0.16         0.84 11.9127639
4          1  y   4    0.20         0.80 19.9504706
5          1  y   5    0.32         0.68 40.7799651
6          1  y   6    0.52         0.48 76.7159181

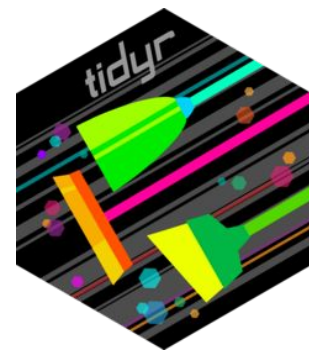
> dim(e3)
[1] 320  6
```

Scenario 1: Exercise 3

- Back in your groups with your new assignments (Base R, tidyr, data.table) work on exercise 3 in the examples.R file.

Scenario 1: Exercise 3

- Report back with your answers/progress



Scenario 1: Exercise 3

- Great, you have the data! But it's in long format and you want it in wide. Widen the data using **tidyr**, **data.table**, and base R.

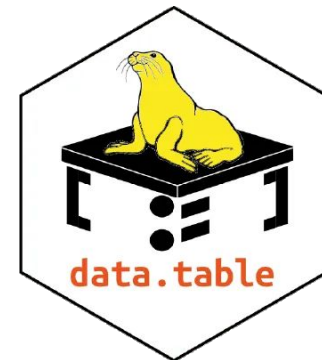
```
> #* `tidyr`  
> library(tidyr)  
> wide<-pivot_wider(e2.5)  
Error in `pivot_wider()`:  
! Can't subset columns that don't exist.  
✖ Column `name` doesn't exist.  
Run `rlang::last_trace()` to see where the error occurred.
```

Solution Next

Scenario 1: Exercise 3 - Solution

- Great, you have the data! But it's in long format and you want it in wide. Widen the data using `tidyr`, `data.table`, and base R.

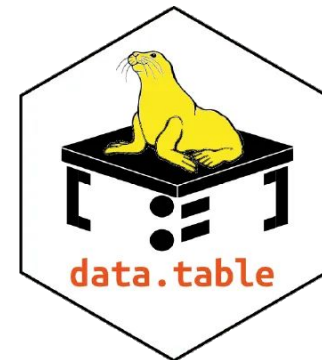
```
> library(tidyr)
> wide<-tidyr::pivot_wider(e2.5, names_from = variable,
+                           values_from = value)
> e3 <- as.data.frame(wide)
```



Scenario 1: Exercise 3

- Great, you have the data! But it's in long format and you want it in wide. Widen the data using tidyr, `data.table`, and base R.

```
> #* `data.table`  
> library(data.table)  
> e2dt <- as.data.table(e2.5)  
> e3 <- dcast(e2dt, experiment ~ variable)  
Aggregate function missing, defaulting to 'length'  
> e3 <- as.data.frame(e3)
```



Scenario 1: Exercise 3

- Great, you have the data! But it's in long format and you want it in wide. Widen the data using tidyr, [data.table](#), and base R.

```
> #* `data.table`  
> library(data.table)  
> e2dt <- as.data.table(e2.5)  
> e3 <- dcast(e2dt, experiment ~ variable)  
Aggregate function missing, defaulting to 'length'  
> e3 <- as.data.frame(e3)  
  
> head(e3)  
  experiment germ_pct hazard_pct mean_area  
1           1      80         80        80  
2           2      80         80        80  
3           3      80         80        80  
4           4      80         80        80
```

Solution Next

Scenario 1: Exercise 3 - Solution

- Great, you have the data! But it's in long format and you want it in wide. Widen the data using `tidyr`, `data.table`, and base R.

```
> library(data.table)
> e2dt <- as.data.table(e2.5)
> wide <- dcast(e2dt, experiment+trt+DAP ~ variable)
> e3 <- as.data.frame(wide)
> head(e3)
```

	experiment	trt	DAP	germ_pct	hazard_pct	mean_area
1	1	j	1	0.00	1.00	0.2621603
2	1	j	2	0.16	0.84	7.6518142
3	1	j	3	0.32	0.68	23.2291706



Scenario 1: Exercise 3

- Great, you have the data! But it's in long format and you want it in wide. Widen the data using `tidyr`, `data.table`, and `base R`.

```
> e3<-e2.5[e2.5$variable=="germ_pct", c("experiment", "trt", "DAP")]
> e3[,3:5]<-lapply(unique(e2.5$variable), function(i) e2.5[e2.5$variable==i, "variable"])
> colnames(e3)<-c(colnames(e3)[1:3], unique(e2.5$variable))
Error in names(x) <- value :
  'names' attribute [6] must be the same length as the vector [5]
```

Note: this is not generally how you'd do this. It works but people who really want to keep R light still tend to do this with `reshape2`.



Scenario 1: Exercise 3

- Great, you have the data! But it's in long format and you want it in wide. Widen the data using `tidyr`, `data.table`, and `base R`.

```
> e3<-e2.5[e2.5$variable=="germ_pct", c("experiment", "trt", "DAP")]
> e3[,3:5]<-lapply(unique(e2.5$variable), function(i) e2.5[e2.5$variable==i, "variable"])
> colnames(e3)<-c(colnames(e3)[1:3], unique(e2.5$variable))
```

```
Error in names(x) <- value :
  'names' attribute [6] must be the same length as the vector [5]
```

```
> head(e3)
  experiment trt      DAP      V4      V5
4          1   y germ_pct hazard_pct mean_area
7          1   y germ_pct hazard_pct mean_area
10         1   y germ_pct hazard_pct mean_area
13         1   y germ_pct hazard_pct mean_area
16         1   y germ_pct hazard_pct mean_area
19         1   y germ_pct hazard_pct mean_area
```



Scenario 1: Exercise 3

- Great, you have the data! But it's in long format and you want it in wide. Widen the data using `tidyr`, `data.table`, and `base R`.

```
> e3<-e2.5[e2.5$variable=="germ_pct", c("experiment", "trt", "DAP")]
> e3[,3:5]<-lapply(unique(e2.5$variable), function(i) e2.5[e2.5$variable==i, "variable"])
> colnames(e3)<-c(colnames(e3)[1:3], unique(e2.5$variable))
Error in names(x) <- value :
  'names' attribute [6] must be the same length as the vector [5]
```

```
> head(e3)
  experiment trt   DAP      V4      V5
4          1   y germ_pct hazard_pct mean_area
7          1   y germ_pct hazard_pct mean_area
10         1   y germ_pct hazard_pct mean_area
13         1   y germ_pct hazard_pct mean_area
16         1   y germ_pct hazard_pct mean_area
19         1   y germ_pct hazard_pct mean_area
```

Solution Next

Scenario 1: Exercise 3 - Solution

- Great, you have the data! But it's in long format and you want it in wide. Widen the data using `tidyr`, `data.table`, and `base R`.

```
> e3<-e2.5[e2.5$variable=="germ_pct", c("experiment", "trt", "DAP")]
> e3[,4:6]<-lapply(unique(e2.5$variable), function(i) e2.5[e2.5$variable==i, "value"])
> colnames(e3)<-c(colnames(e3)[1:3], unique(e2.5$variable))
> head(e3)
```

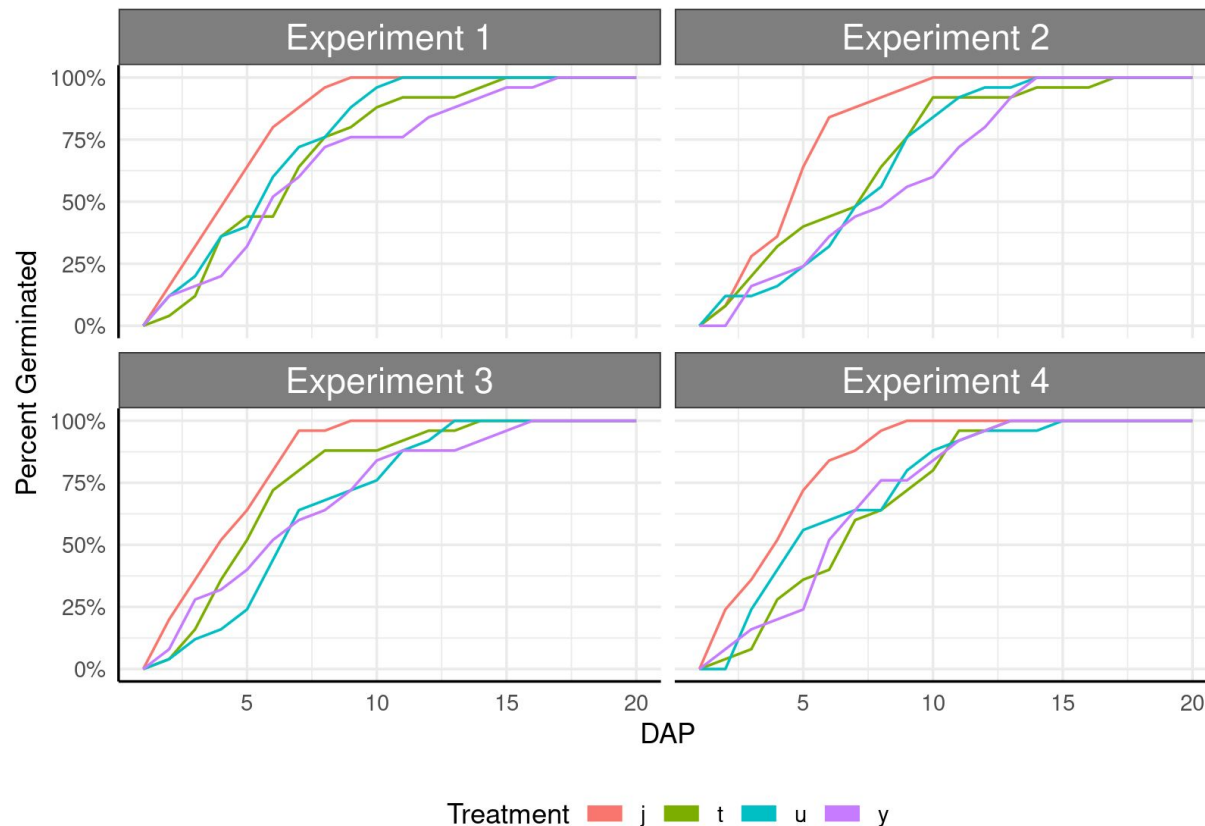
	experiment	trt	DAP	germ_pct	hazard_pct	mean_area
4	1	y	1	0.00	1.00	0.1120951
7	1	y	2	0.12	0.88	6.0952852
10	1	y	3	0.16	0.84	11.9127639
13	1	y	4	0.20	0.80	19.9504706
16	1	y	5	0.32	0.68	40.7799651
19	1	y	6	0.52	0.48	76.7159181

Take a break

- Come back sometime

Scenario 1: Exercise 4

- You need to visualize the data now, but ggplot is throwing errors. Troubleshoot the code and get to a plot like this (don't sweat the theme elements too much)



Exercise 4 – ggplot2 reminder

- ggplot objects are made in layers and are stored as lists.
- Among other things this means that you can troubleshoot a ggplot line by line by just commenting out the ``+`` at the end of lines as you go through the code.

Scenario 1: Exercise 4

- We write out a first try at our ggplot and see what happens
- Work in your groups to troubleshoot this figure

```
> ggplot(e3, aes(x=DAP, y=germ_pct, group = experiment))+  
+   facet_wrap(experiment)+  
+   geom_line(color=trt)+  
+   scale_y_continuous(labels=percent_format())+  
+   guides(color=guide_legend(override.aes=list(linewidth=3)))+  
+   labs(color="Treatment", y="Percent Germinated")+  
+   theme(legend.position="below")
```

Scenario 1: Exercise 4

- Predictably we get an error. Starting with a smaller version of the plot would have made this easier.

```
> ggplot(e3, aes(x=DAP, y=germ_pct, group = experiment))+  
+   facet_wrap(experiment)+  
+   geom_line(color=trt)+  
+   scale_y_continuous(labels=percent_format())+  
+   guides(color=guide_legend(override.aes=list(linewidth=3)))+  
+   labs(color="Treatment", y="Percent Germinated")+  
+   theme(legend.position="below")  
Error: object 'experiment' not found
```

Scenario 1: Exercise 4

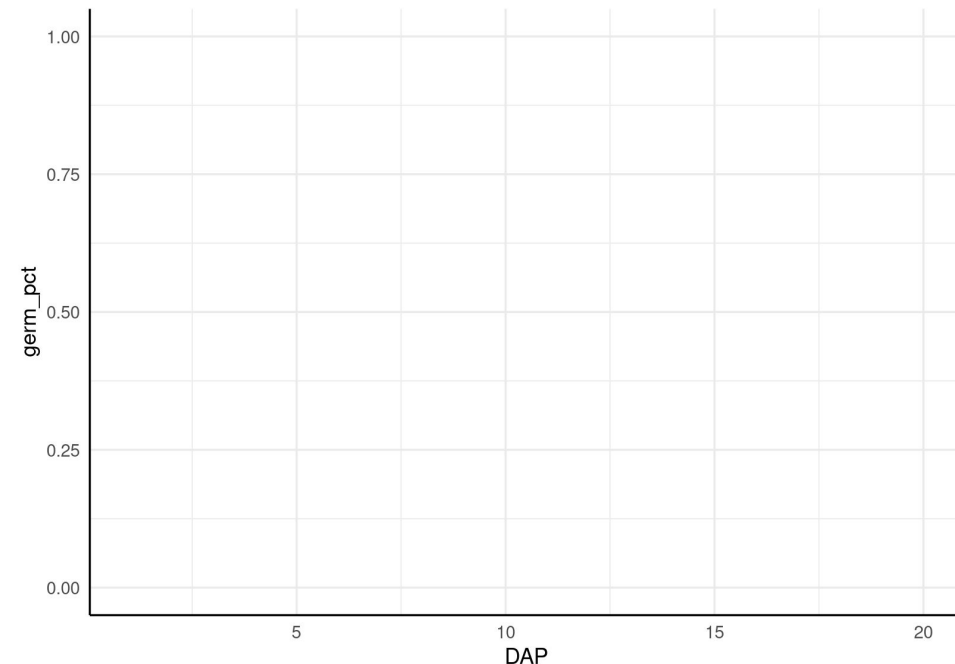
- We say “experiment” twice in this ggplot but we can run the first line on it’s own to see if the error persists.

```
> ggplot(e3, aes(x=DAP, y=germ_pct, group = experiment))+  
+   facet_wrap(experiment)+  
+   geom_line(color=trt)+  
+   scale_y_continuous(labels=percent_format())+  
+   guides(color=guide_legend(override.aes=list(linewidth=3)))+  
+   labs(color="Treatment", y="Percent Germinated")+  
+   theme(legend.position="below")  
Error: object 'experiment' not found
```

Scenario 1: Exercise 4

- We say “experiment” twice in this ggplot but we can run the first line on it’s own to see if the error persists.
 - It does not, so we found the problem area.

```
> ggplot(e3, aes(x=DAP, y=germ_pct, group = experiment))#+  
> |
```



Scenario 1: Exercise 4

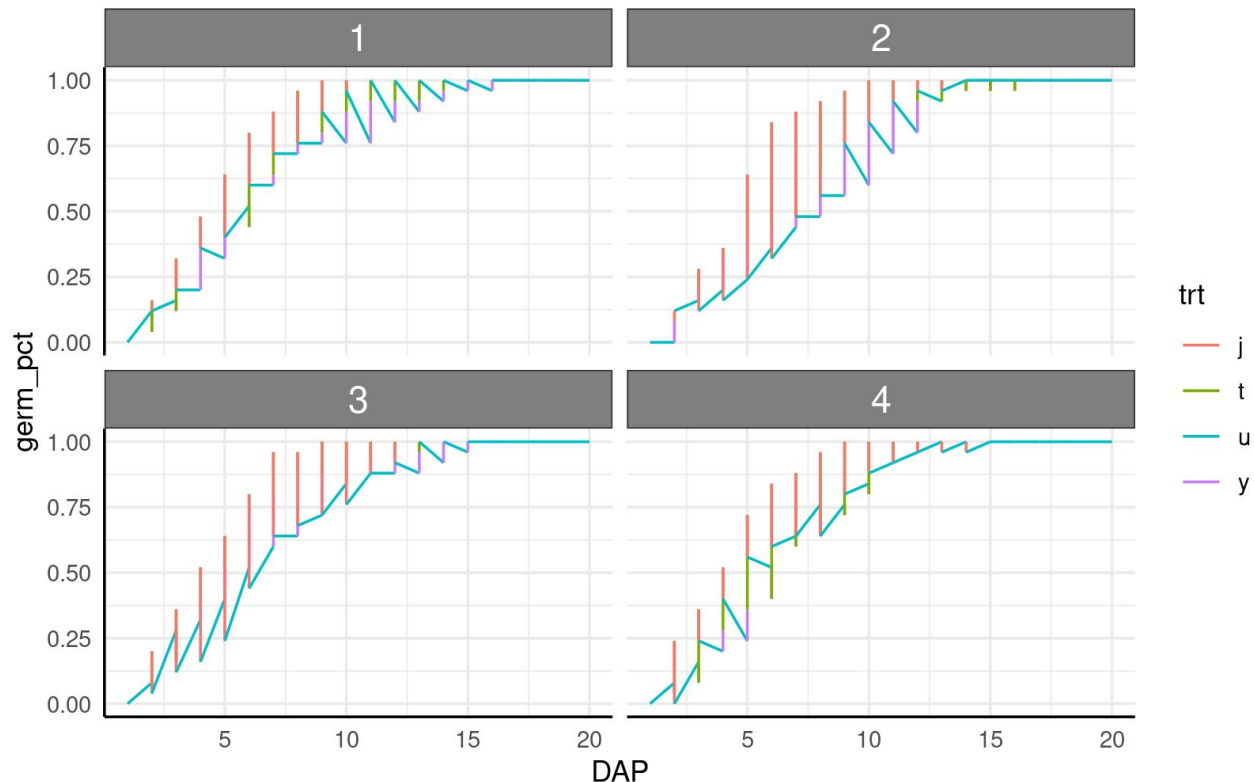
- We check the docs, fix the faceting and keep going, but we hit another problem.

```
> ggplot(e3, aes(x=DAP, y=germ_pct, group = experiment))+  
+   facet_wrap(~experiment)+  
+   geom_line(color=trt)#+  
Error: object 'trt' not found
```

Scenario 1: Exercise 4

- We check the docs, fix the aesthetic and keep going without errors, but our plot looks weird.

```
> ggplot(e3, aes(x=DAP, y=germ_pct, group = experiment))+  
+   facet_wrap(~experiment)+  
+   geom_line(aes(color=trt))#+  
> |
```

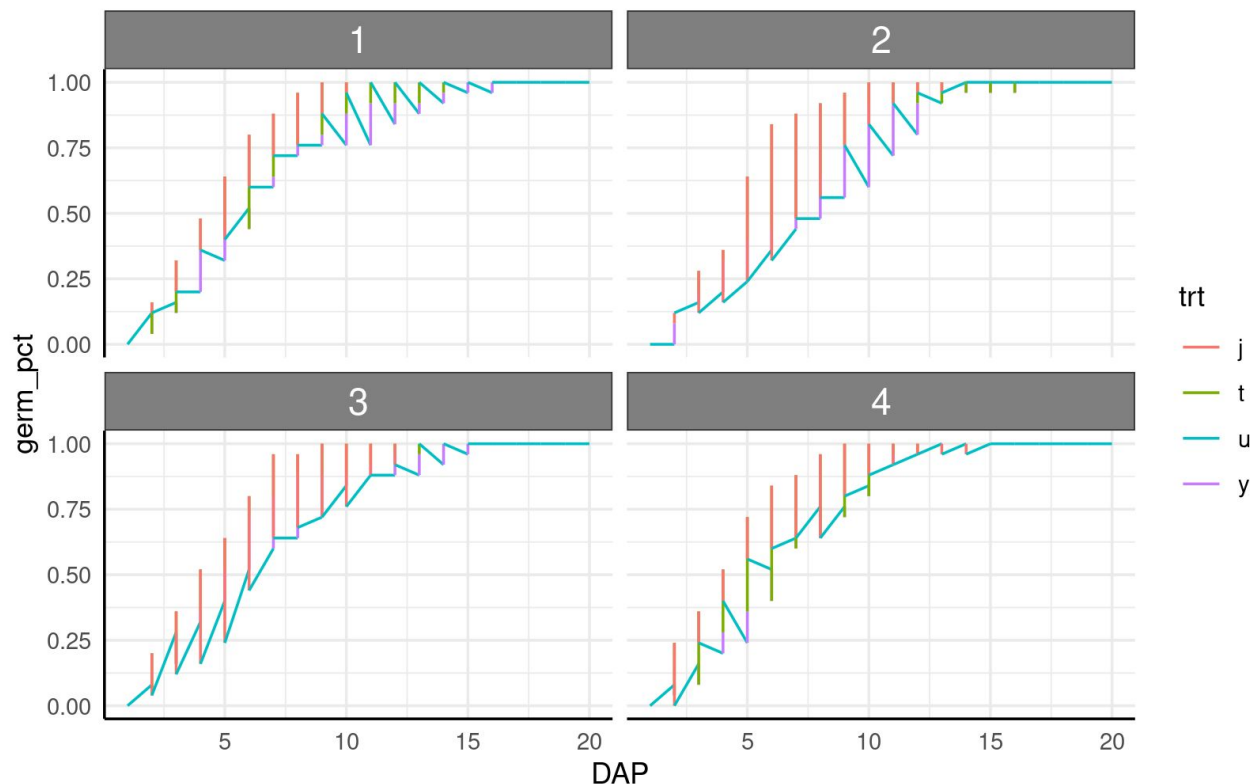


Scenario 1: Exercise 4

- We check the docs, fix the aesthetic and keep going without errors, but our plot looks weird.

```
> ggplot(e3, aes(x=DAP, y=germ_pct, group = experiment))+  
+   facet_wrap(~experiment)+  
+   geom_line(aes(color=trt))#+  
>
```

- We have too much data at each timepoint. We should be tracking individuals, so we check the grouping.

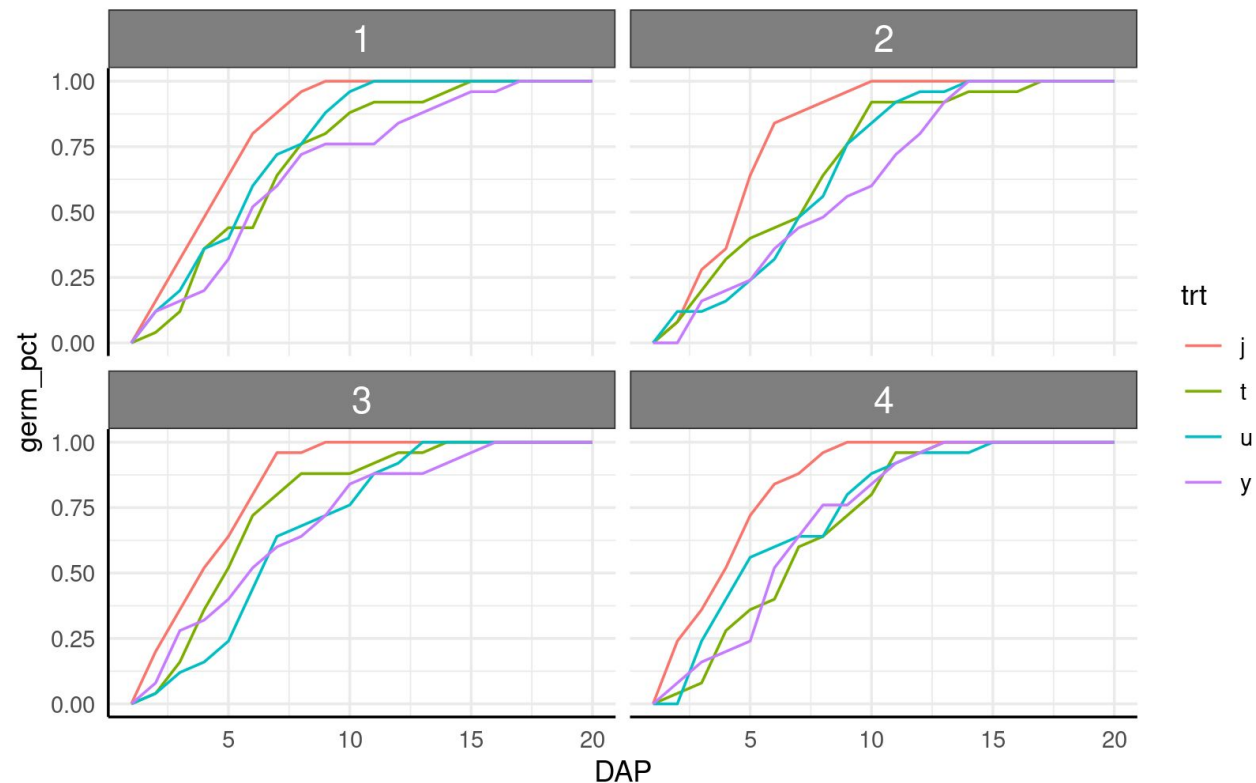


Scenario 1: Exercise 4

- After diagnosing the problem and looking at our data we see we need to specify treatment in our grouping.

```
> ggplot(e3, aes(x=DAP, y=germ_pct, group = interaction(experiment,trt)))+  
+   facet_wrap(~experiment)+  
+   geom_line(aes(color=trt))#+  
>
```

- Much better!



Scenario 1: Exercise 4

- We keep going and get a new kind of error.

```
> ggplot(e3, aes(x=DAP, y=germ_pct, group = interaction(experiment,trt)))+  
+   facet_wrap(~experiment)+  
+   geom_line(aes(color=trt))+  
+   scale_y_continuous(labels=percent_format())#+  
Error in percent_format() : could not find function "percent_format"
```

Scenario 1: Exercise 4

- We keep going and get a new kind of error.
 - When a function isn't found `??` can be useful

```
> ggplot(e3, aes(x=DAP, y=germ_pct, group = interaction(experiment,trt)))+  
+   facet_wrap(~experiment)+  
+   geom_line(aes(color=trt))+  
+   scale_y_continuous(labels=percent_format())#+  
Error in percent_format() : could not find function "percent_format"
```

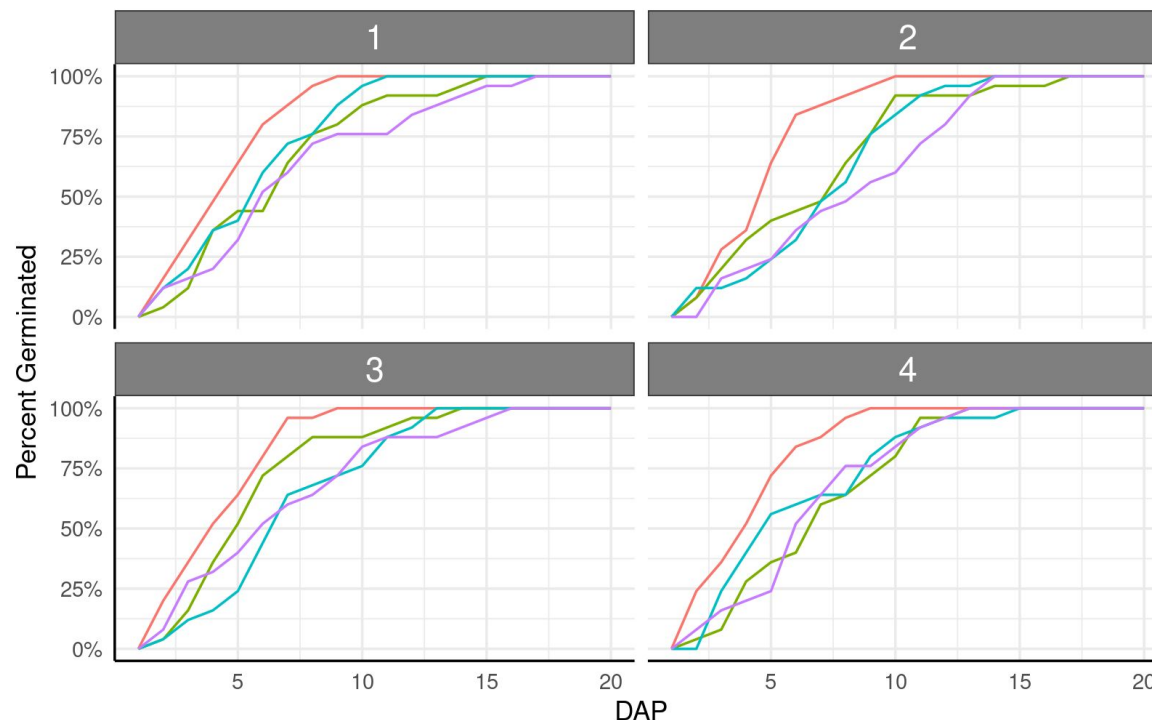
```
> ??percent_format
```

Help pages:

[scales::percent_format](#) Superseded interface to 'label_percent()'

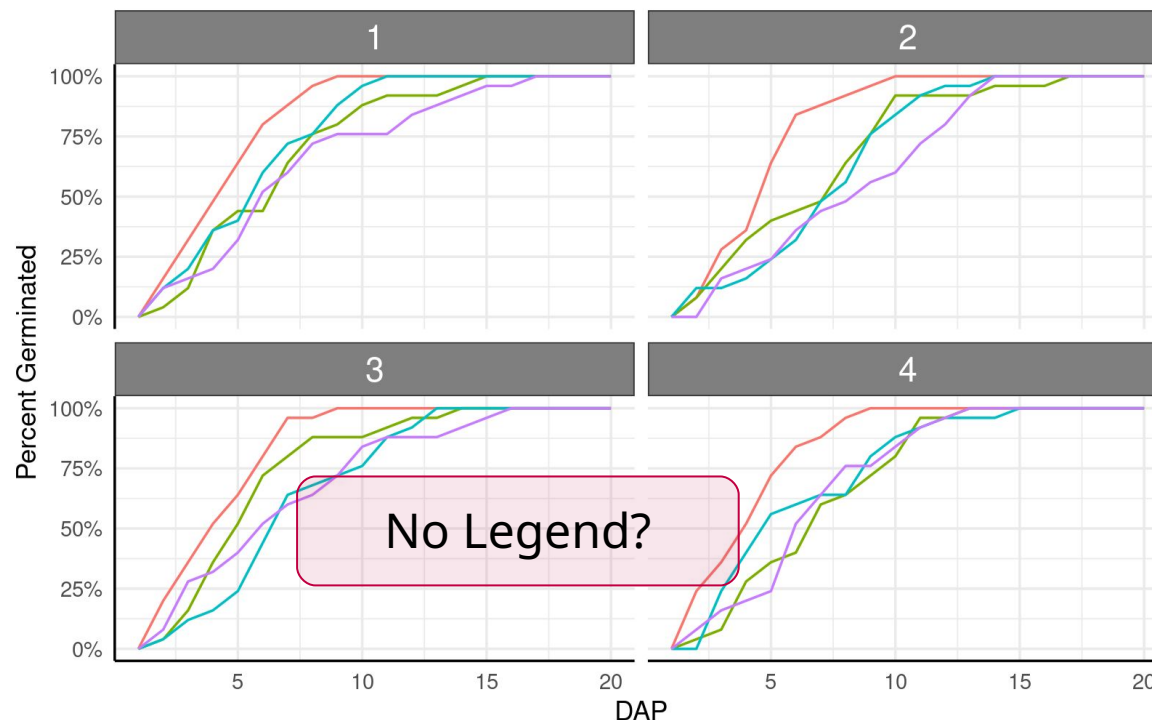
Scenario 1: Exercise 4

```
> ggplot(e3, aes(x=DAP, y=germ_pct, group = interaction(experiment,trt)))+  
+   facet_wrap(~experiment)+  
+   geom_line(aes(color=trt))+  
+   scale_y_continuous(labels=scales::label_percent())+  
+   guides(color=guide_legend(override.aes=list(linewidth=3)))+  
+   labs(color="Treatment", y="Percent Germinated")+  
+   theme(legend.position="below")
```



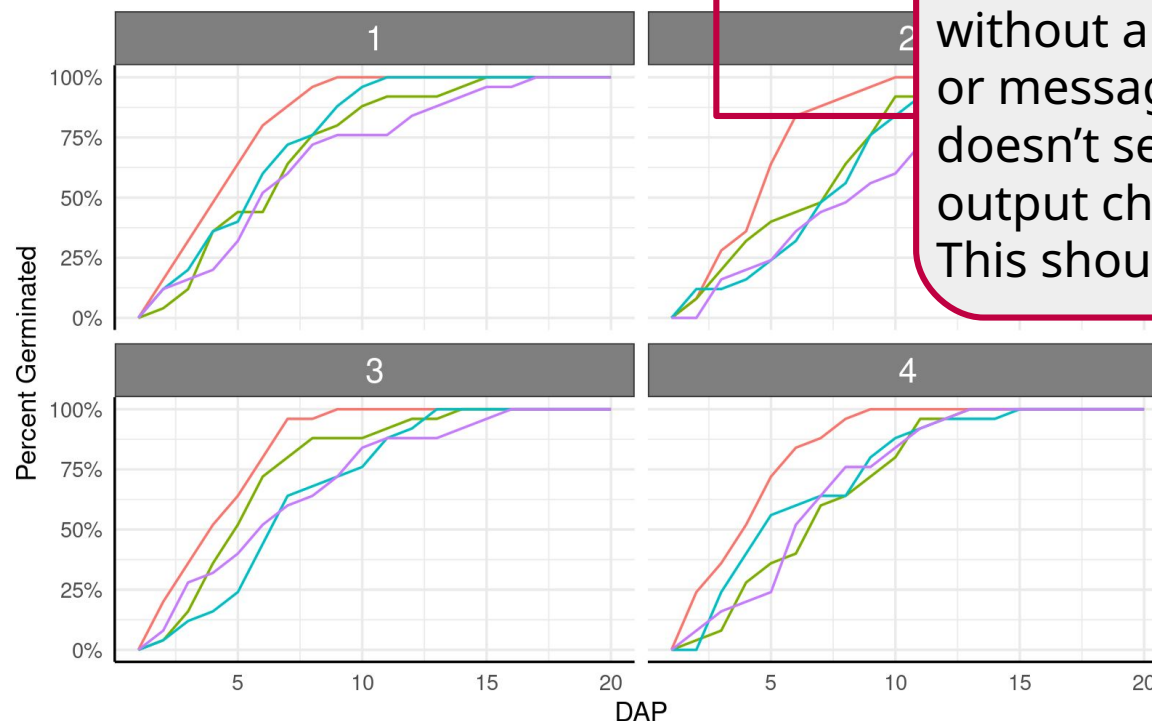
Scenario 1: Exercise 4

```
> ggplot(e3, aes(x=DAP, y=germ_pct, group = interaction(experiment,trt)))+  
+   facet_wrap(~experiment)+  
+   geom_line(aes(color=trt))+  
+   scale_y_continuous(labels=scales::label_percent())+  
+   guides(color=guide_legend(override.aes=list(linewidth=3)))+  
+   labs(color="Treatment", y="Percent Germinated")+  
+   theme(legend.position="below")
```



Scenario 1: Exercise 4

```
> ggplot(e3, aes(x=DAP, y=germ_pct, group = interaction(experiment,trt)))+  
+   facet_wrap(~experiment)+  
+   geom_line(aes(color=trt))+  
+   scale_y_continuous(labels=scales::label_percent())+  
+   guides(color=guide_legend(override.aes=list(linewidth=3)))+  
+   labs(color="Treatment", y="Percent Germinated")+  
+   theme(legend.position="below")
```

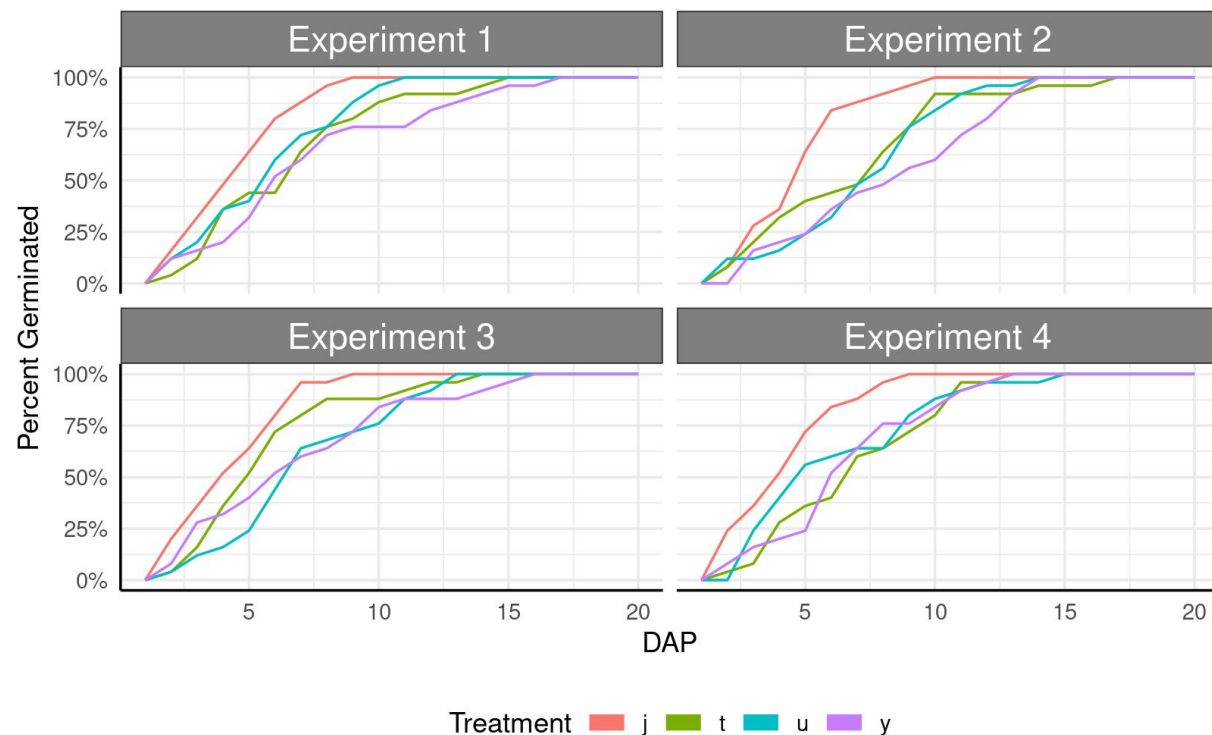


Some things can “fail” without an error, warning, or message. When a function doesn’t seem to change your output check the docs. Here This should be “bottom”.

Solution Next

Scenario 1: Exercise 4 - Solution

```
> ggplot(e3, aes(x=DAP, y=germ_pct, group = interaction(experiment, trt)))+  
+   facet_wrap(~paste0("Experiment ", experiment))+  
+   geom_line(aes(color=trt))+  
+   scale_y_continuous(labels=scales::label_percent())+  
+   guides(color=guide_legend(override.aes=list(linewidth=3)))+  
+   labs(color="Treatment", y="Percent Germinated")+  
+   theme(legend.position="bottom")  
>
```



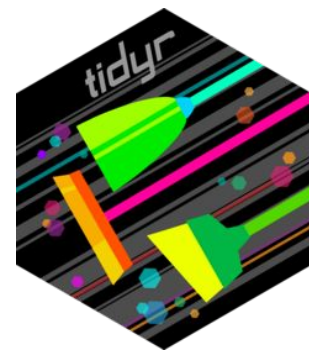
- Does anyone have other takeaways from working on that ggplot?
 - Other questions?
- Optionally we can take a break here.

Scenario 1: Exercise 5

- Plots are looking good! So good that your PI wants you to make some statistical comparisons. Problem is that this data has all 25 reps per treatment per experiment collapsed to a rate.
 - You need to take the data from the percentage space to a binary “germinated” or “non Germinated” status per plant per timepoint, then filter for only plants that germinated or were censored (ungerminated on last day)
 - Should end with 5818 rows.
 - Again, using `tidyr`, `data.table`, and base R.
 - In your groups work with your assigned variety of R.

Scenario 1: Exercise 5

- Report back on your answers/progress



Scenario 1: Exercise 5

- You know each treatment:experiment combination had 25 plants, expand the percentage to a binary status per plant using `tidyr`.

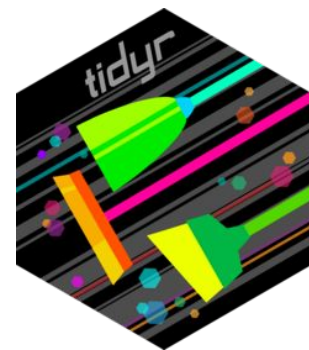
```
> t5 <- e3%>%  
+   mutate(nGerm = 25 * germ_pct, nZero = 25 - nGerm)%>%  
+   pivot_longer(cols = c(nGerm, nZero))%>%  
+   uncount(value)%>%  
+   mutate(germ = ifelse(name=="nZero", 0, 1))%>%  
+   select(experiment, trt, DAP, germ)
```

Error in `uncount()`:

! Can't convert from `weights` <double> to <integer> due to loss of precision.

• Locations: 177, 178, 245, 295, 296, 325, 527, 609, 610

Run ``rlang::last_trace()`` to see where the error occurred.



Scenario 1: Exercise 5

- You know each treatment:experiment combination had 25 plants, expand the percentage to a binary status per plant using `tidyr`.

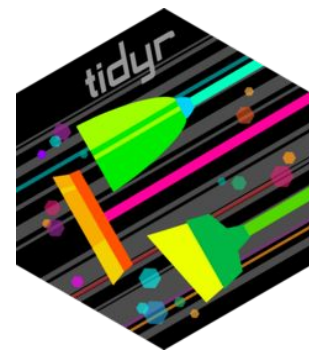
Error in ``uncount()``:

! Can't convert from ``weights`` `<double>` to `<integer>` due to loss of precision.

• Locations: 177, 178, 245, 295, 296, 325, 527, 609, 610

Run ``rlang::last_trace()`` to see where the error occurred.

- This has a useful hint about what is wrong, even if we don't know much about the ``uncount()`` function
 - Converting double to integer is causing a problem.



Data Types

- Doubles are like more “granular” numbers than integers.

Error in ``uncount()``:

! Can't convert from ``weights`` `<double>` to `<integer>` due to loss of precision.

• Locations: 177, 178, 245, 295, 296, 325, 527, 609, 610

Run ``rlang::last_trace()`` to see where the error occurred.

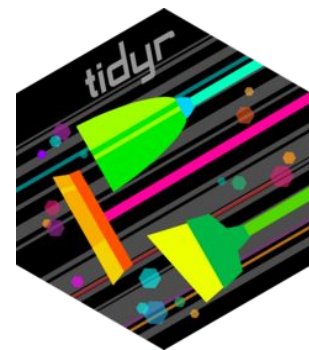
```
> 2==2
```

```
[1] TRUE
```

```
> as.double(2)==as.integer(2)
```

```
[1] TRUE
```

Pop Quiz:
What is 0.56×25 ?



Data Types

- Doubles are like more “granular” numbers than integers.

Error in ``uncount()``:

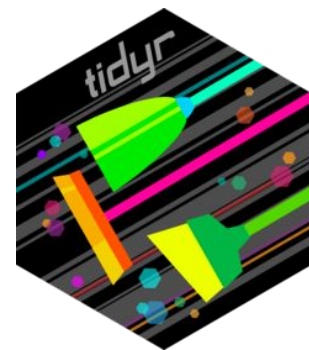
! Can't convert from ``weights`` `<double>` to `<integer>` due to loss of precision.

• Locations: 177, 178, 245, 295, 296, 325, 527, 609, 610

Run ``rlang::last_trace()`` to see where the error occurred.

```
> 2==2
[1] TRUE
> as.double(2)==as.integer(2)
[1] TRUE
> 0.56*25==14
[1] FALSE
> as.integer(0.56*25)==14
[1] TRUE
> round(0.56*25)==14
[1] TRUE
> #pain
```

Solution Next

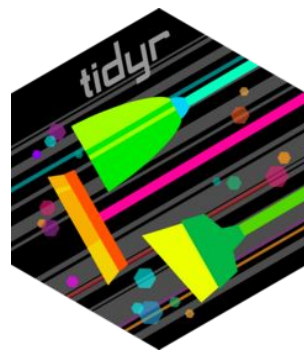


Scenario 1: Exercise 5

- You know each treatment:experiment combination had 25 plants, expand the percentage to a binary status per plant using `tidyr`.

```
> t5 <- e3%>%  
+   mutate(nGerm = 25 * germ_pct, nZero = 25 - nGerm)%>%  
+   pivot_longer(cols = c(nGerm, nZero))%>%  
+   uncount(round(value))%>%  
+   mutate(germ = ifelse(name=="nZero", 0, 1))%>%  
+   select(experiment, trt, DAP, germ)  
> table(as.numeric(table(t5$experiment, t5$DAP, t5$trt)))
```

```
25  
320  
> |
```



Scenario 1: Exercise 5

- You know each treatment:experiment combination had 25 plants, expand the percentage to a binary status per plant using `tidyr`.

```
> t5 <- t5 %>%  
+   filter(germ == 1 | DAP == max(t5$DAP))
```

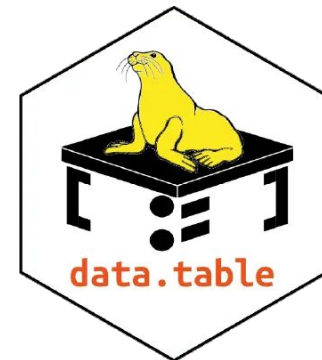
```
> head(t5,2)
```

```
# A tibble: 2 × 4
```

	experiment	trt	DAP	germ
	<i><int></i>	<i><chr></i>	<i><int></i>	<i><dbl></i>
1	1	y	2	1
2	1	y	2	1

```
> dim(t5)
```

```
[1] 5818 4
```

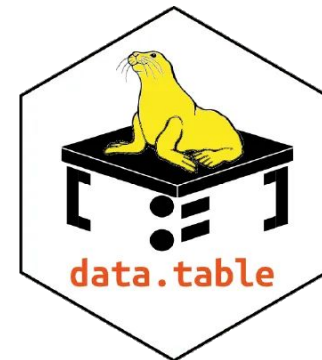


Scenario 1: Exercise 5

- You know each treatment:experiment combination had 25 plants, expand the percentage to a binary status per plant using `data.table`.

```
> library(data.table)
> d5 <- as.data.table(e3)
> d5[, nGerm := 25*germ_pct ]
> d5[, nZero := 25-nGerm ]
> d5 <- melt(d5, id.vars = c("experiment"), measure.vars = "nGerm")
> d5 <- d5[rep(1:.N, value),c(1:3)]
> d5[,germ := ifelse(variable=="nGerm", 1, 0)]
> head(d5)
```

	experiment	variable	value	germ
1:	1	nGerm	3	1
2:	1	nGerm	3	1
3:	1	nGerm	3	1
4:	1	nGerm	4	1
5:	1	nGerm	4	1
6:	1	nGerm	4	1



Scenario 1: Exercise 5

- You know each treatment:experiment combination had 25 plants, expand the percentage to a binary status per plant using `data.table`.

```
> library(data.table)
> d5 <- as.data.table(e3)
> d5[, nGerm := 25*germ_pct ]
> d5[, nZero := 25-nGerm ]
> d5 <- melt(d5, id.vars = c("experiment"), measure.vars = "nGerm")
> d5 <- d5[rep(1:.N, value),c(1:3)]
> d5[,germ := ifelse(variable=="nGerm", 1, 0)]
> head(d5)
```

	experiment	variable	value	germ
1:	1	nGerm	3	1
2:	1	nGerm	3	1
3:	1	nGerm	3	1
4:	1	nGerm	4	1
5:	1	nGerm	4	1
6:	1	nGerm	4	1

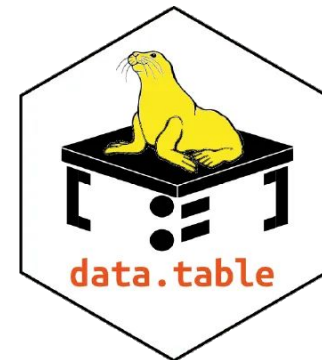
```
> dim(t5)
```

```
[1] 8000
```

```
> dim(d5)
```

```
[1] 5818
```

- Something is wrong here.
 - $25 * 4 * 4 * 20 = 8000$
 - Find and fix the problem

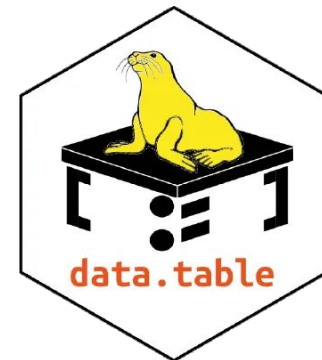


Scenario 1: Exercise 5

- Data table has some different syntax and can modify objects “in place”. This can make it harder to find the problem area.

```
> library(data.table)
> d5 <- as.data.table(e3)
> d5[, nGerm := 25*germ_pct ]
> d5[, nZero := 25-nGerm ]
> d5 <- melt(d5, id.vars = c("experiment"), measure.vars = "nGerm")
> d5 <- d5[rep(1:.N, value),c(1:3)]
> d5[,germ := ifelse(variable=="nGerm", 1, 0)]
> head(d5)
```

	experiment	variable	value	germ
1:	1	nGerm	3	1
2:	1	nGerm	3	1
3:	1	nGerm	3	1
4:	1	nGerm	4	1
5:	1	nGerm	4	1
6:	1	nGerm	4	1

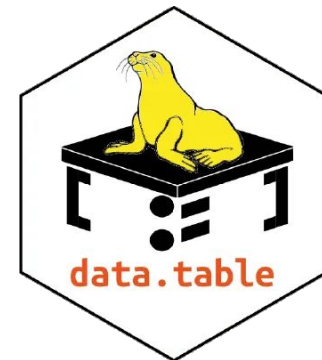


Scenario 1: Exercise 5

- We check the contents of d5 before melting and it looks okay, so we dig into the `melt` more.
 - We could also have looked at what step is most error-prone and checked before and after that first.

```
> d5 <- as.data.table(e3)
> d5[, nGerm := 25*germ_pct ]
> d5[, nZero := 25-nGerm ]
> head(d5)
```

	experiment	trt	DAP	germ_pct	hazard_pct	mean_area	nGerm	nZero
1:	1	y	1	0.00	1.00	0.1120951	0	25
2:	1	y	2	0.12	0.88	6.0952852	3	22



Scenario 1: Exercise 5

- We check the contents of d5 before melting and it looks okay, so we dig into the `melt` more.

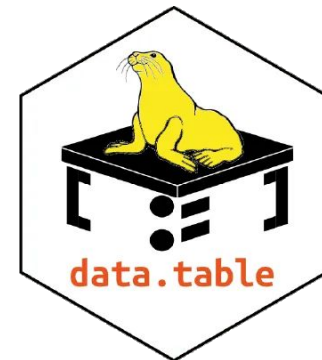
```
> d5 <- melt(d5, id.vars = c("experiment"), measure.vars = "nGerm")  
> head(d5)
```

	experiment	variable	value
1:	1	nGerm	0
2:	1	nGerm	3
3:	1	nGerm	4
4:	1	nGerm	5
5:	1	nGerm	8
6:	1	nGerm	13

```
> table(d5$variable)
```

```
nGerm  
320
```

- This should have nGerm and nZero as variables and we lost a lot of metadata, so we found the first problem area.



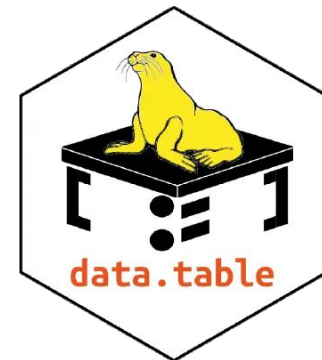
Scenario 1: Exercise 5

- Now our melt is working, but we still get an error

```
> d5 <- melt(d5, id.vars = c("experiment", "trt", "DAP"), measure.vars = c("nGerm","nZero"))
> head(d5,2)
  experiment trt DAP variable value
1:          1  y   1    nGerm     0
2:          1  y   2    nGerm     3
> table(d5$variable)

nGerm nZero
  320   320

> d5 <- as.data.table(e3)
> d5[, nGerm := 25*germ_pct ]
> d5[, nZero := 25-nGerm ]
> d5 <- melt(d5, id.vars = c("experiment", "trt", "DAP"), measure.vars = c("nGerm","nZero"))
> d5 <- d5[rep(1:.N, value),c(1:3)]
> d5[,germ := ifelse(variable=="nGerm", 1, 0)]
Error in eval(jsub, SEnv, parent.frame()) : object 'variable' not found
```

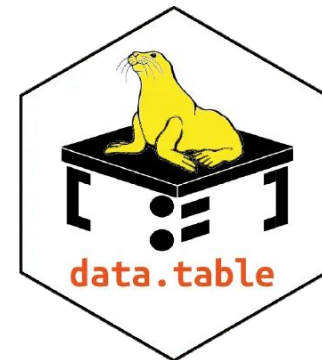


Scenario 1: Exercise 5

- `object “variable” not found` means it isn't in the data, so we check our subsetting and keep the `variable` column.

```
> d5 <- as.data.table(e3)
> d5[, nGerm := 25*germ_pct ]
> d5[, nZero := 25-nGerm ]
> d5 <- melt(d5, id.vars = c("experiment", "trt", "DAP"), measure.vars = c("nGerm","nZero"))
> d5 <- d5[rep(1:.N, value),c(1:4)]
> head(d5,2)
```

	experiment	trt	DAP	variable
1:	1	y	2	nGerm
2:	1	y	2	nGerm



Scenario 1: Exercise 5

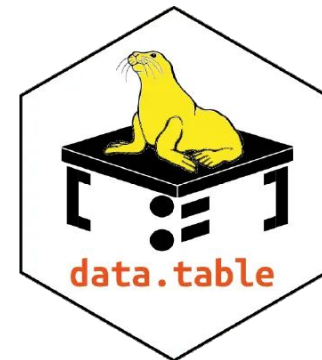
- This looks good now, but we can drop the `variable` column at this point to keep it a little cleaner.

```
> d5 <- as.data.table(e3)
> d5[, nGerm := 25*germ_pct ]
> d5[, nZero := 25-nGerm ]
> d5 <- melt(d5, id.vars = c("experiment", "trt", "DAP"), measure.vars = c("nGerm","nZero"))
> d5 <- d5[rep(1:.N, value),c(1:4)]
> head(d5,2)
```

	experiment	trt	DAP	variable
1:	1	y	2	nGerm
2:	1	y	2	nGerm

```
> d5[,germ := ifelse(variable=="nGerm", 1, 0)]
> head(d5)
```

	experiment	trt	DAP	variable	germ
1:	1	y	2	nGerm	1
2:	1	y	2	nGerm	1

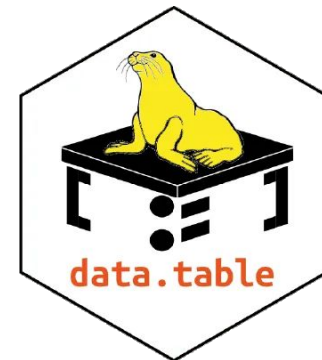


Scenario 1: Exercise 5

- This looks good now, but we can drop the `variable` column at this point to keep it a little cleaner.

```
> d5 <- as.data.table(e3)
> d5[, nGerm := 25*germ_pct ]
> d5[, nZero := 25-nGerm ]
> d5 <- melt(d5, id.vars = c("experiment", "trt", "DAP"), measure.vars = c("nGerm","nZero"))
> d5 <- d5[rep(1:.N, value),c(1:4)]
> d5[,germ := ifelse(variable=="nGerm", 1, 0)]
> d5<-d5[,c(1:3,5)]
> head(d5,2)
```

	experiment	trt	DAP	germ
1:	1	y	2	1
2:	1	y	2	1



Scenario 1: Exercise 5

- If we're a little more careful we notice that we don't uniformly have 25 reps though.
 - Should sound familiar.

```
> d5 <- as.data.table(e3)
> d5[, nGerm := 25*germ_pct ]
> d5[, nZero := 25-nGerm ]
> d5 <- melt(d5, id.vars = c("experiment", "trt", "DAP"), measure.vars = c("nGerm","nZero"))
> d5 <- d5[rep(1:.N, value),c(1:4)]
> d5[,germ := ifelse(variable=="nGerm", 1, 0)]
> d5<-d5[,c(1:3,5)]
> head(d5,2)
   experiment trt DAP germ
1:           1  y   2    1
2:           1  y   2    1
> table(as.numeric(table(d5$experiment, d5$DAP, d5$trt)))
```

24 25
3 317

Solution Next

Scenario 1: Exercise 5

- Again we have an integer vs double problem, but base R's `rep` function does not warn us how `tidyr::uncount` did.

```
> d5 <- as.data.table(e3)
> d5[, nGerm := 25*germ_pct ]
> d5[, nZero := 25-nGerm ]
> d5 <- melt(d5, id.vars = c("experiment", "trt", "DAP"), measure.vars = c("nGerm", "nZero"))
> d5 <- d5[rep(1:.N, round(value)),c(1:4)]
> d5[,germ := ifelse(variable=="nGerm", 1, 0)]
> d5<-d5[,c(1:3,5)]
> head(d5,2)
```

	experiment	trt	DAP	germ
1:	1	y	2	1
2:	1	y	2	1

```
> table(as.numeric(table(d5$experiment, d5$DAP, d5$trt)))
```

```
25
320
```

```
> sum(c(25-(0.56*25), 0.56*25))
[1] 25
> length(rep.int("a", 25-(0.56*25)))
[1] 10
```

- In practice this is a rare problem to run into, but it can be **very** frustrating

Scenario 1: Exercise 5

- Again we have an integer vs double problem, but base R's `rep` function does not warn us how `tidyr::uncount` did.

```
> table(as.numeric(table(d5$experiment, d5$DAP, d5$trt)))
```

```
25
```

```
320
```

```
> dim(d5)
```

```
[1] 8000 4
```

```
>
```

```
> d5 <- d5[germ == 1 | DAP == max(d5$DAP), ]
```

```
> head(d5,2)
```

	experiment	trt	DAP	germ
1:	1	y	2	1
2:	1	y	2	1

```
> dim(d5)
```

```
[1] 5818 4
```



Scenario 1: Exercise 5

- You know each treatment:experiment combination had 25 plants, expand the percentage to a binary status per plant using `base R`.

```
> nPerCondition <- 25
> e5 <- do.call(rbind, lapply(1:nrow(e3), function(i){
+   row <- e3[i,c("experiment", "trt", "DAP", "germ_pct")]
+   nGerm <- row$germ_pct*nPerCondition
+   nNon <- nPerCondition - nGerm
+   x<-data.frame(experiment = row$experiment,
+                 trt = row$trt, DAP = row$DAP,
+                 germ = rep(c(1,0), times = c(nGerm, nNon)))
+   return(x)
+ })))
```



Scenario 1: Exercise 5

- You know each treatment:experiment combination had 25 plants, expand the percentage to a binary status per plant using **base R**.

```
> nPerCondition <- 25
> e5 <- do.call(rbind, lapply(1:nrow(e3), function(i){
+   row <- e3[i,c("experiment", "trt", "DAP", "germ_pct")]
+   nGerm <- row$germ_pct*nPerCondition
+   nNon <- nPerCondition - nGerm
+   x<-data.frame(experiment = row$experiment,
+                 trt = row$trt, DAP = row$DAP,
+                 germ = rep(c(1,0), times = c(nGerm, nNon)))
+   return(x)
+ })))
```



Scenario 1: Exercise 5

- You know each treatment:experiment combination had 25 plants, expand the percentage to a binary status per plant using **base R**.

```
> nPerCondition <- 25
> e5 <- do.call(rbind, lapply(1:nrow(e3), function(i){
+   row <- e3[i,c("experiment", "trt", "DAP", "germ_pct")]
+   nGerm <- row$germ_pct*nPerCondition
+   nNon <- nPerCondition - nGerm
+   x<-data.frame(experiment = row$experiment,
+                 trt = row$trt, DAP = row$DAP,
+                 germ = rep(c(1,0), times = c(nGerm, nNon)))
+   return(x)
+ })))
```



Scenario 1: Exercise 5

- You know each treatment:experiment combination had 25 plants, expand the percentage to a binary status per plant using **base R**.

```
> nPerCondition <- 25
> e5 <- do.call(rbind, lapply(1:nrow(e3), function(i){
+   row <- e3[i,c("experiment", "trt", "DAP", "germ_pct")]
+   nGerm <- row$germ_pct*nPerCondition
+   nNon <- nPerCondition - nGerm
+   x<-data.frame(experiment = row$experiment,
+                 trt = row$trt, DAP = row$DAP,
+                 germ = rep(c(1,0), times = c(nGerm, nNon)))
+   return(x)
+ })))
> table(as.numeric(table(e5$experiment, e5$DAP, e5$trt)))
```

24	25
3	317

Solution Next

Scenario 1: Exercise 5 - Solution

- Now you're used to one of the most confusing errors!

```
> nPerCondition <- 25
> e5 <- do.call(rbind, lapply(1:nrow(e3), function(i){
+   row <- e3[i,c("experiment", "trt", "DAP", "germ_pct")]
+   nGerm <- row$germ_pct*nPerCondition
+   nNon <- nPerCondition - nGerm
+   x<-data.frame(experiment = row$experiment,
+                 trt = row$trt, DAP = row$DAP,
+                 germ = rep(c(1,0), times = c(round(nGerm), round(nNon))))
+   return(x)
+ })))
> table(as.numeric(table(e5$experiment, e5$DAP, e5$trt)))
```

```
25
320
> head(e5,2)
  experiment trt DAP germ
1          1  y   1    0
2          1  y   1    0
```

Scenario 1: Exercise 5 - Solution

- Now you're used to one of the most confusing errors!
- Last step is an easy filter.

```
> table(as.numeric(table(e5$experiment, e5$DAP, e5$trt)))
```

```
25
```

```
320
```

```
>
```

```
> e5 <- e5[e5$germ == 1 | e5$DAP == max(e5$DAP), ]
```

```
> head(e5,2)
```

	experiment	trt	DAP	germ
26	1	y	2	1
27	1	y	2	1

```
> dim(e5)
```

```
[1] 5818 4
```

Let's take a break

- Really go wild
 - Maybe do some math on paper and enjoy how predictable your integers are
 - Just do what you'd like for the next 5-10 minutes.

Scenario 1: Exercise 6

- Now we have what is called “time to event data” and we are ready to make a model and look at our treatment effect on germination.

Scenario 1: Exercise 6

- Now we have what is called “time to event data” and we are ready to make a model and look at our treatment effect on germination.
- For lack of knowing better, we try making a linear model. How can we assess this model?

```
> m6 <- lm(germ ~ DAP*trt, data = e5)
```

Scenario 1: Exercise 6

```
> m6 <- lm(germ ~ DAP*trt, data = e5)
> summary(m6)
```

Call:

```
lm(formula = germ ~ DAP * trt, data = e5)
```

Residuals:

	Min	1Q	Median	3Q	Max
	-3.000e-14	0.000e+00	0.000e+00	0.000e+00	5.813e-12

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	1.000e+00	5.098e-15	1.962e+14	< 2e-16	***
DAP	-4.290e-29	3.870e-16	0.000e+00	1.00000	
trtt	3.032e-28	7.908e-15	0.000e+00	1.00000	
trtu	6.198e-29	7.981e-15	0.000e+00	1.00000	
trty	3.461e-14	8.125e-15	4.259e+00	2.08e-05	***
DAP:trtt	-5.733e-29	5.845e-16	0.000e+00	1.00000	
DAP:trtu	-3.228e-29	5.881e-16	0.000e+00	1.00000	
DAP:trty	-2.288e-15	5.946e-16	-3.848e+00	0.00012	***

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

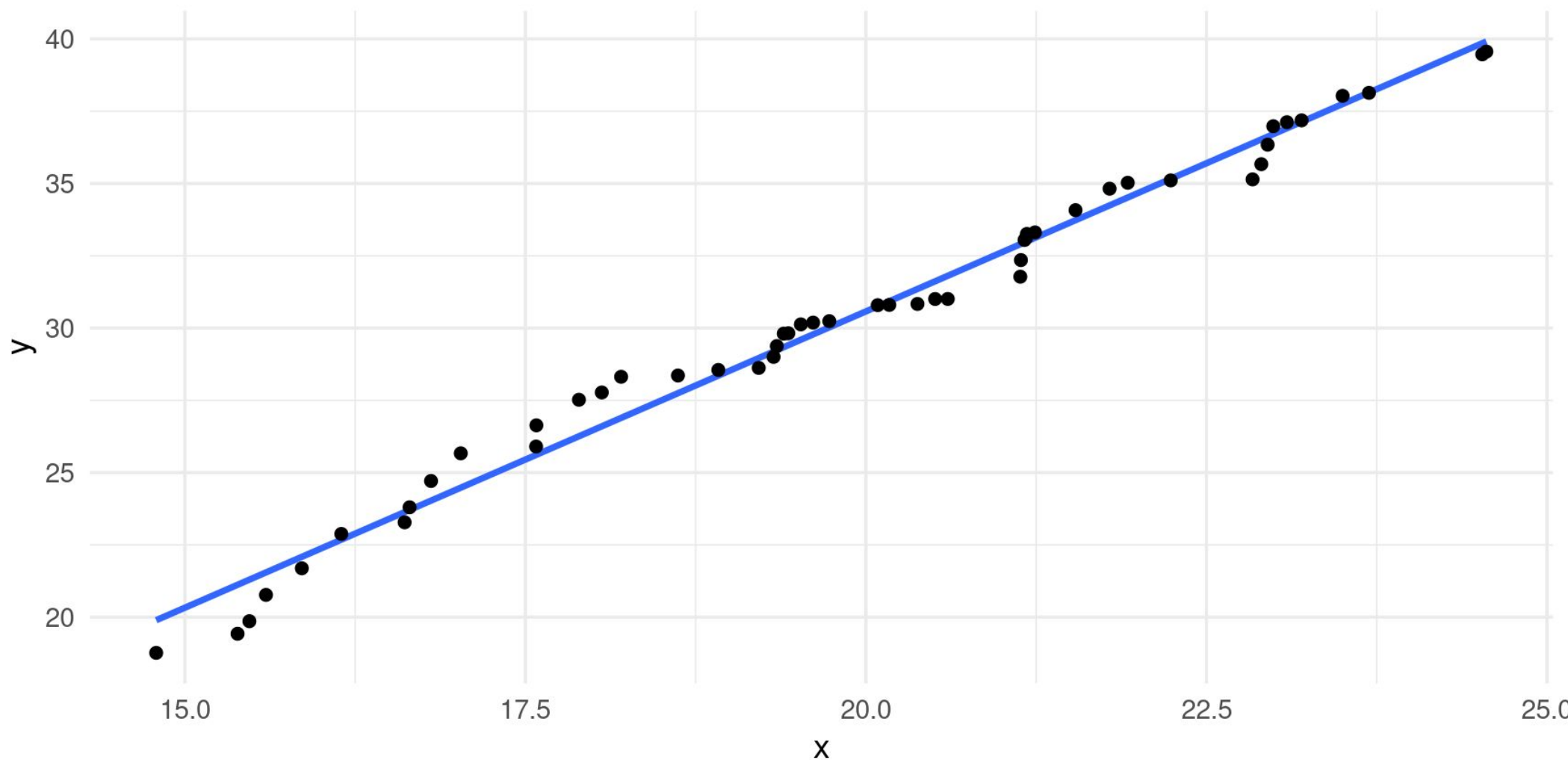
Residual standard error: 7.646e-14 on 5810 degrees of freedom

Multiple R-squared: 0.5, Adjusted R-squared: 0.4994

F-statistic: 830 on 7 and 5810 DF, p-value: < 2.2e-16

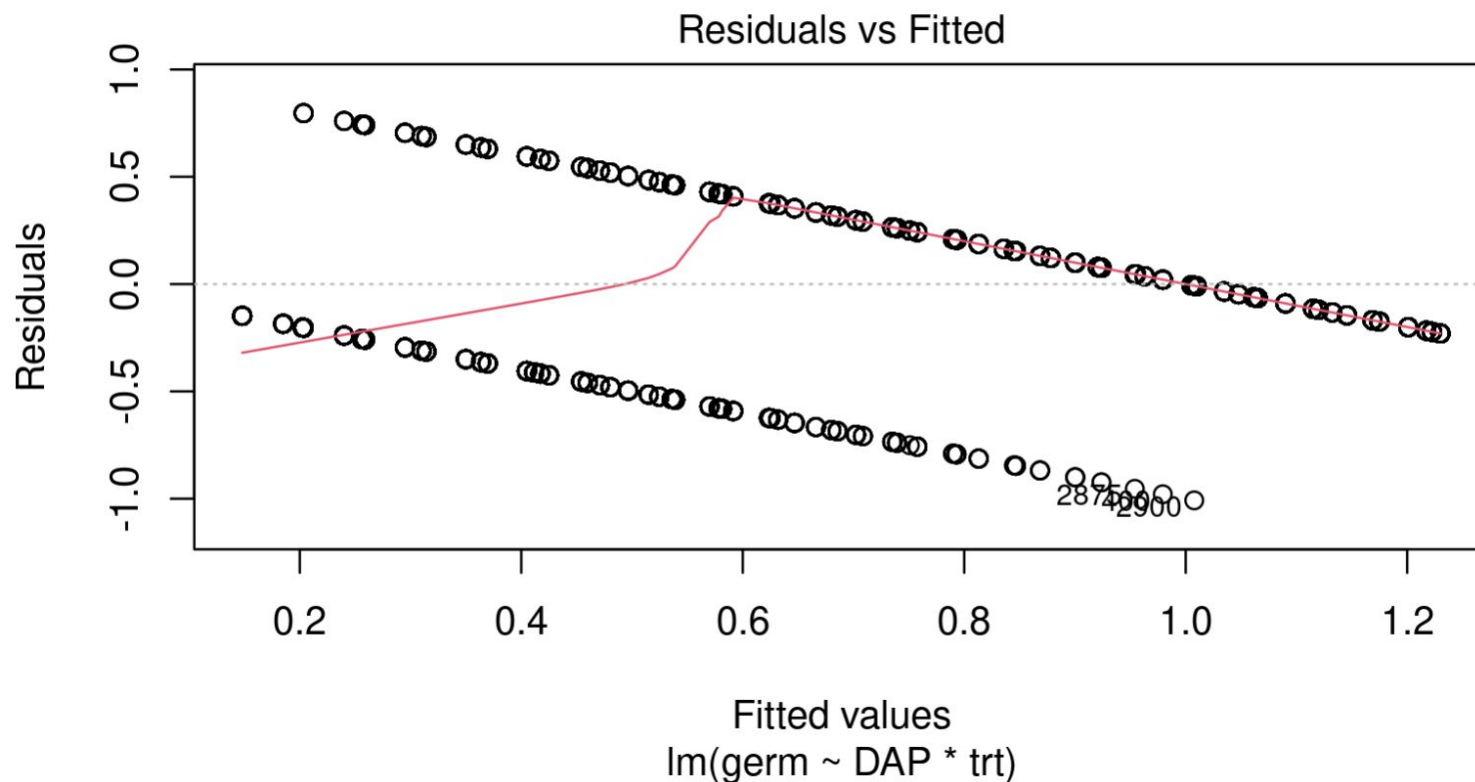
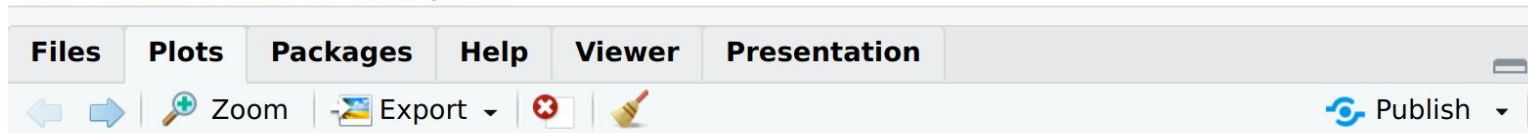
Scenario 1: Exercise 6

Example of a good linear model to compare against

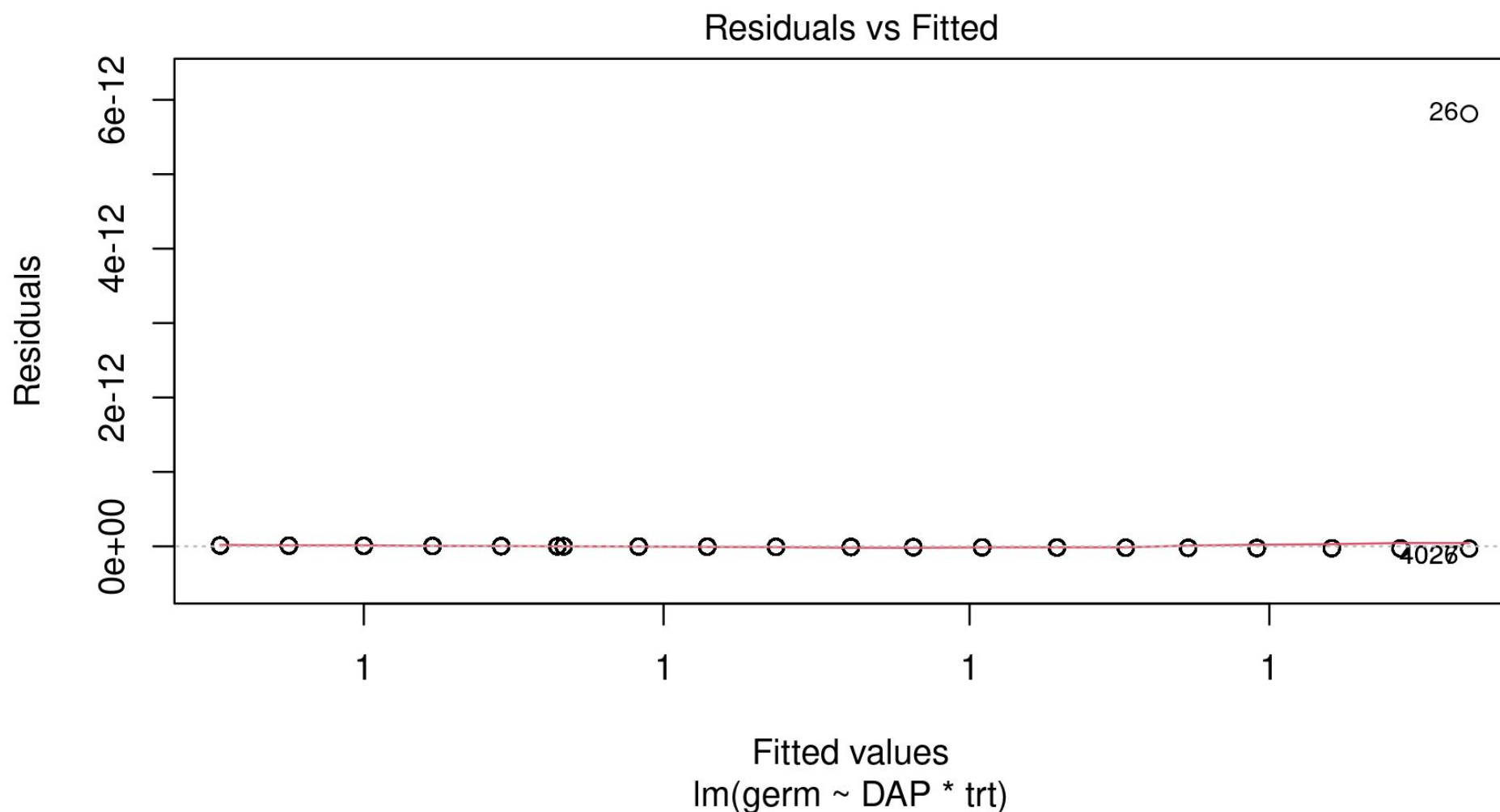


Scenario 1: Exercise 6 – no filters

```
> plot(m6)  
Hit <Return> to see next plot:  
Hit <Return> to see next plot:  
Hit <Return> to see next plot:  
Hit <Return> to see next plot:
```

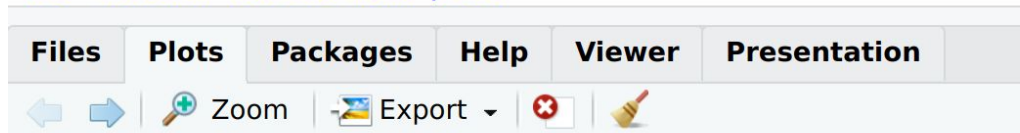


Scenario 1: Exercise 6 – with filters

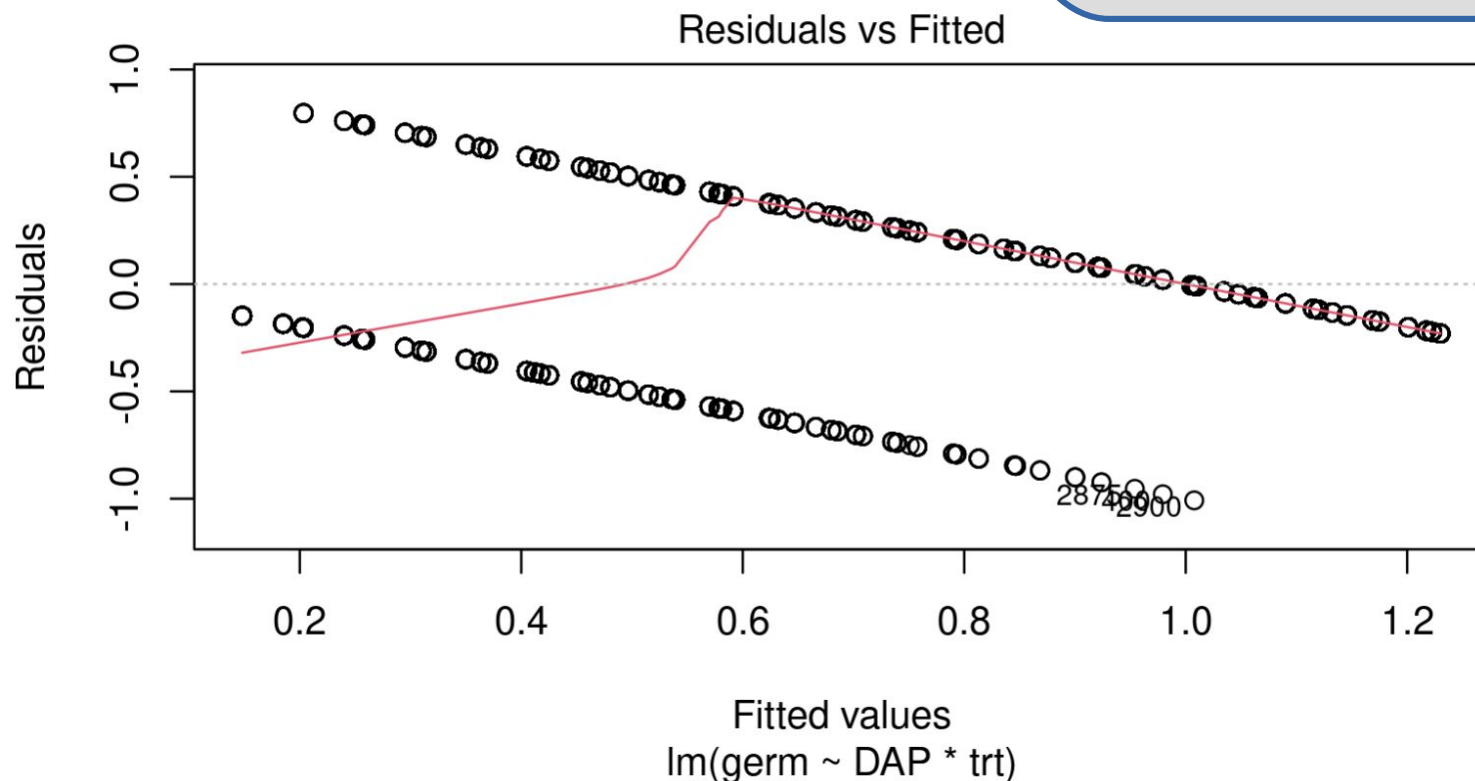


Scenario 1: Exercise 6

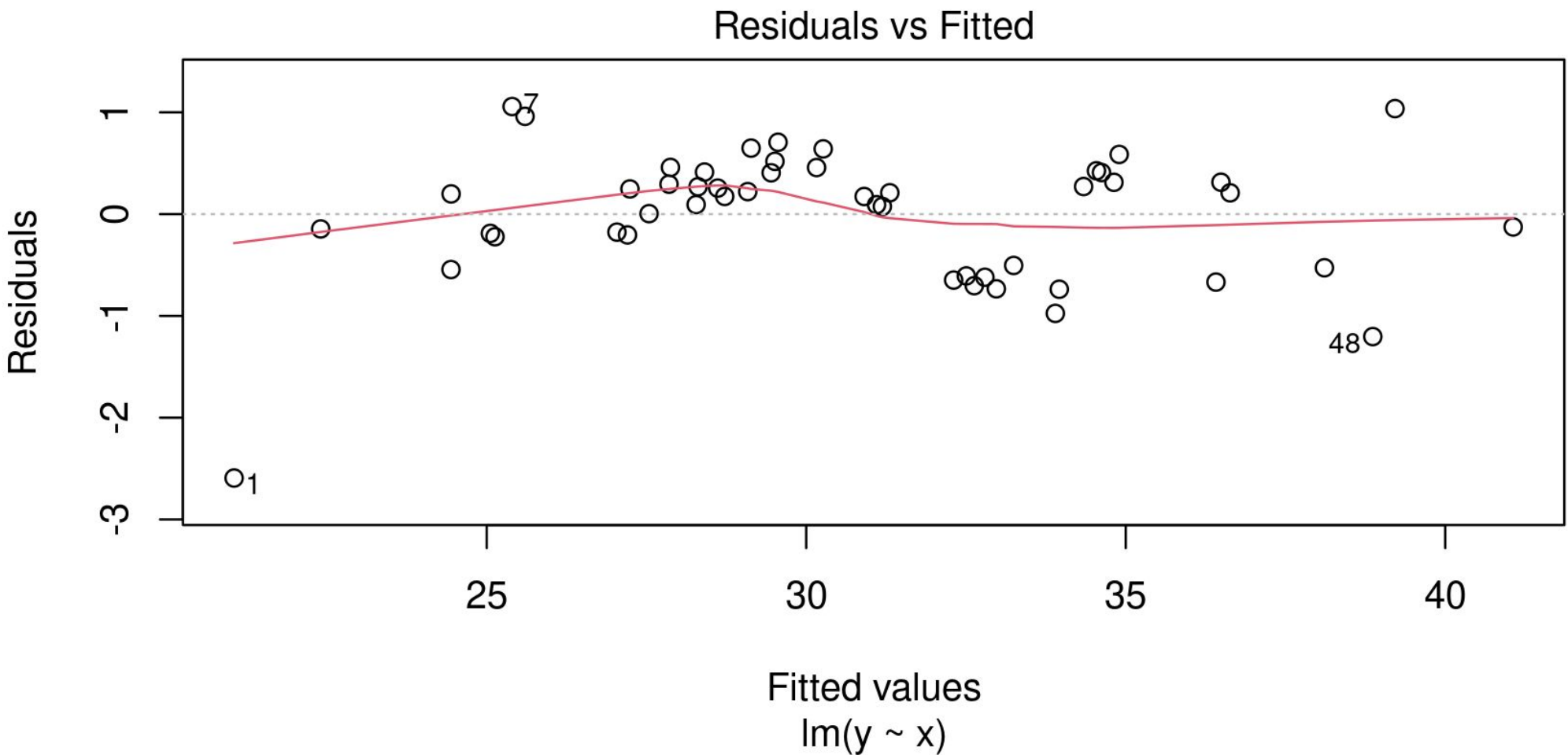
```
> plot(m6)  
Hit <Return> to see next plot:  
Hit <Return> to see next plot:  
Hit <Return> to see next plot:  
Hit <Return> to see next plot:
```



Residuals vs Fitted plots should generally show that residuals are uncorrelated to the fitted values. Here we have a strong pattern and should think about The model carefully.

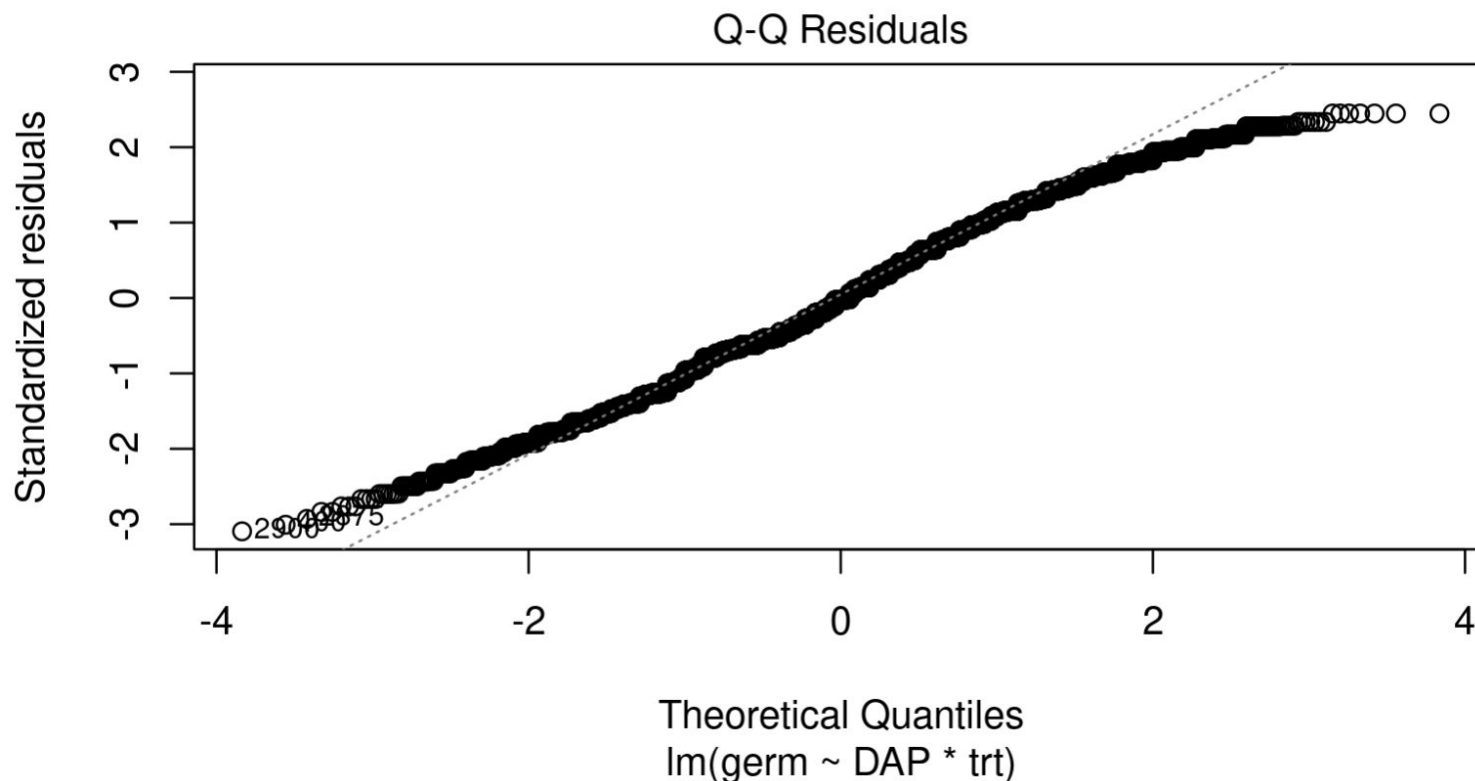
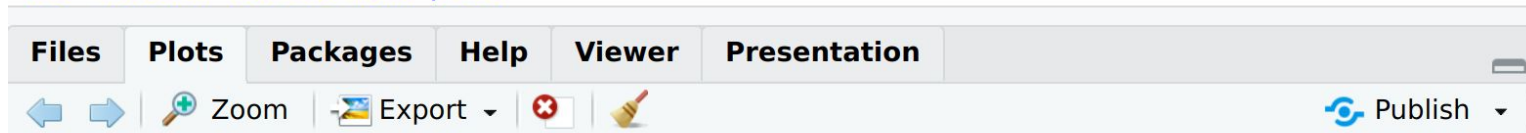


Scenario 1: Exercise 6 - GOOD

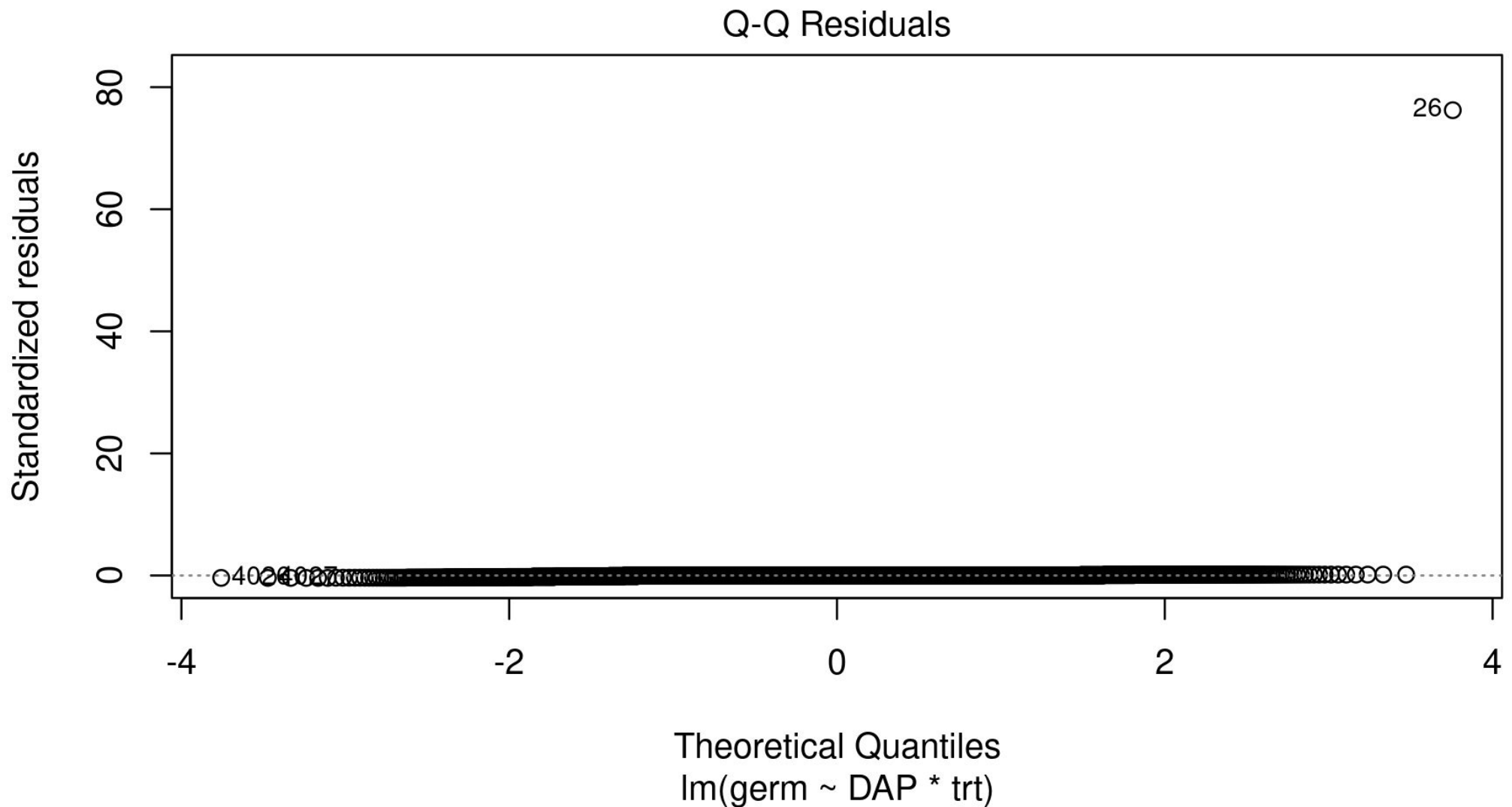


Scenario 1: Exercise 6 – no filters

```
> plot(m6)  
Hit <Return> to see next plot:  
Hit <Return> to see next plot:  
Hit <Return> to see next plot:  
Hit <Return> to see next plot:
```

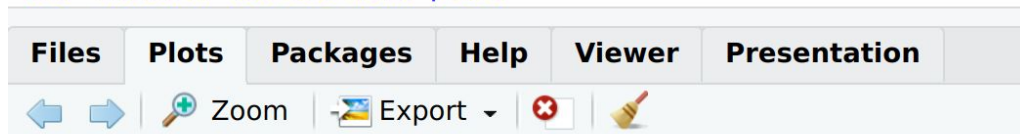


Scenario 1: Exercise 6 – with filters

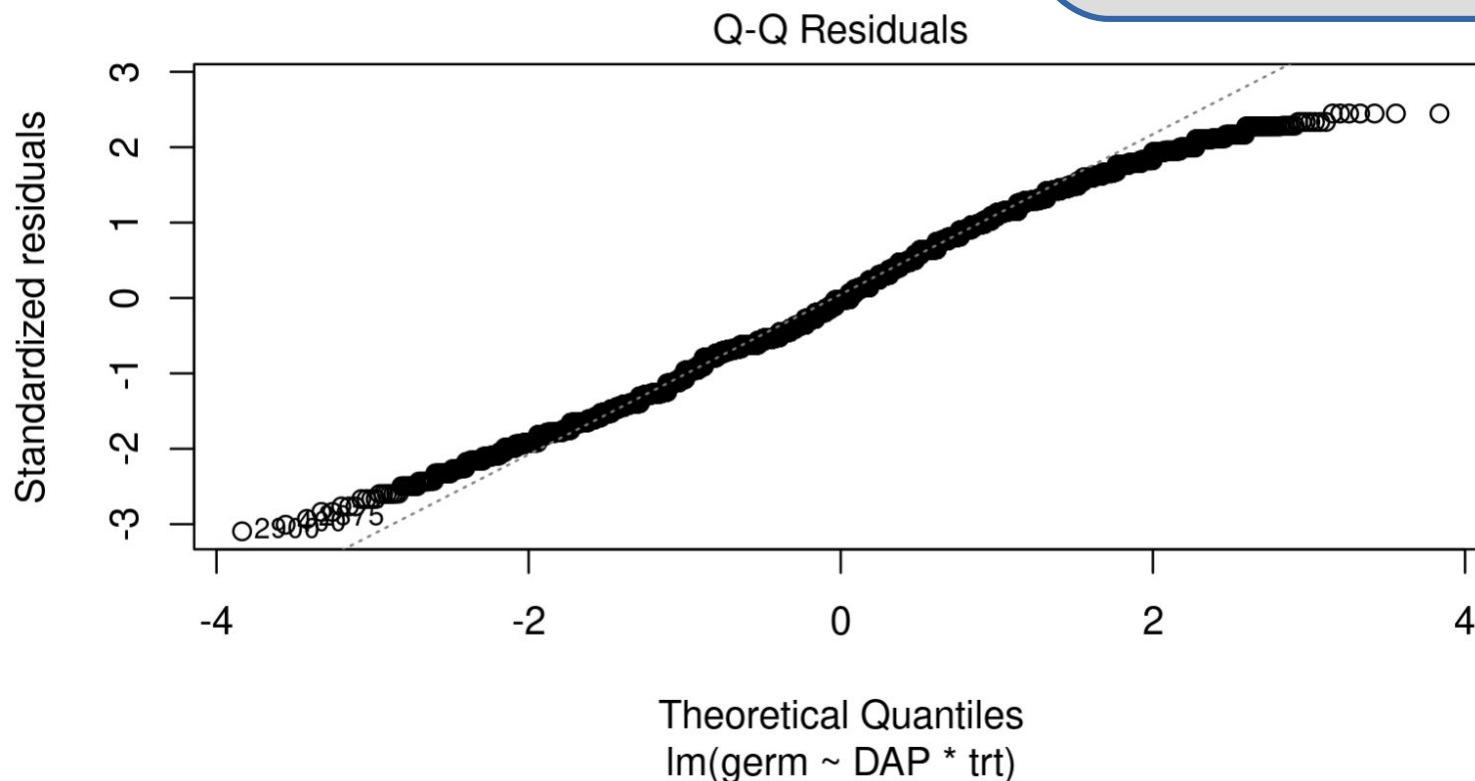


Scenario 1: Exercise 6

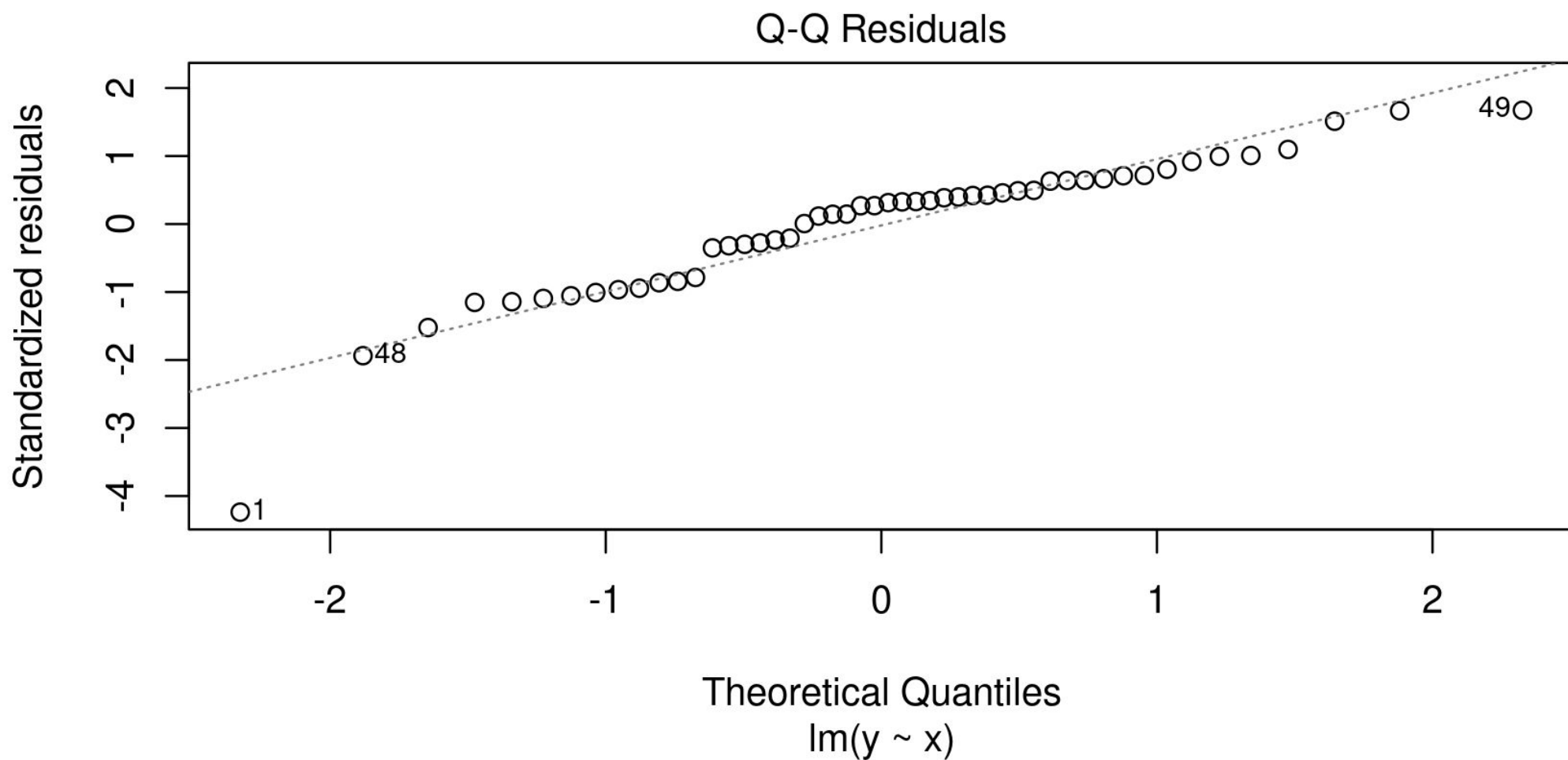
```
> plot(m6)  
Hit <Return> to see next plot:  
Hit <Return> to see next plot:  
Hit <Return> to see next plot:  
Hit <Return> to see next plot:
```



The Q-Q plot is useful for looking at whether or not the model's residuals are normally distributed. Here they don't look too bad, but we can see some drift near the extremes.

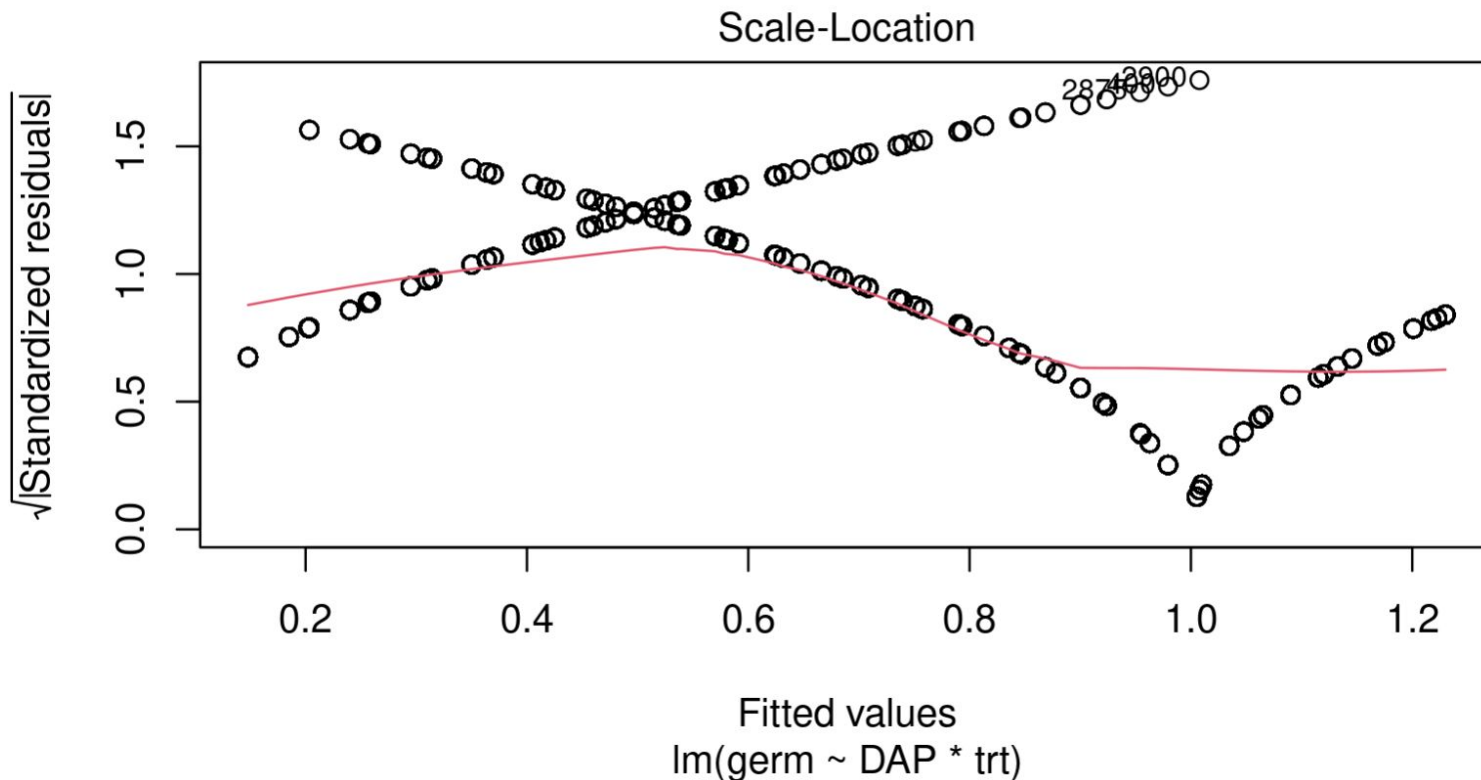
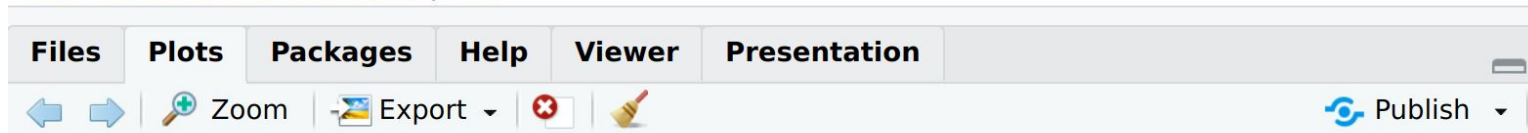


Scenario 1: Exercise 6 - GOOD

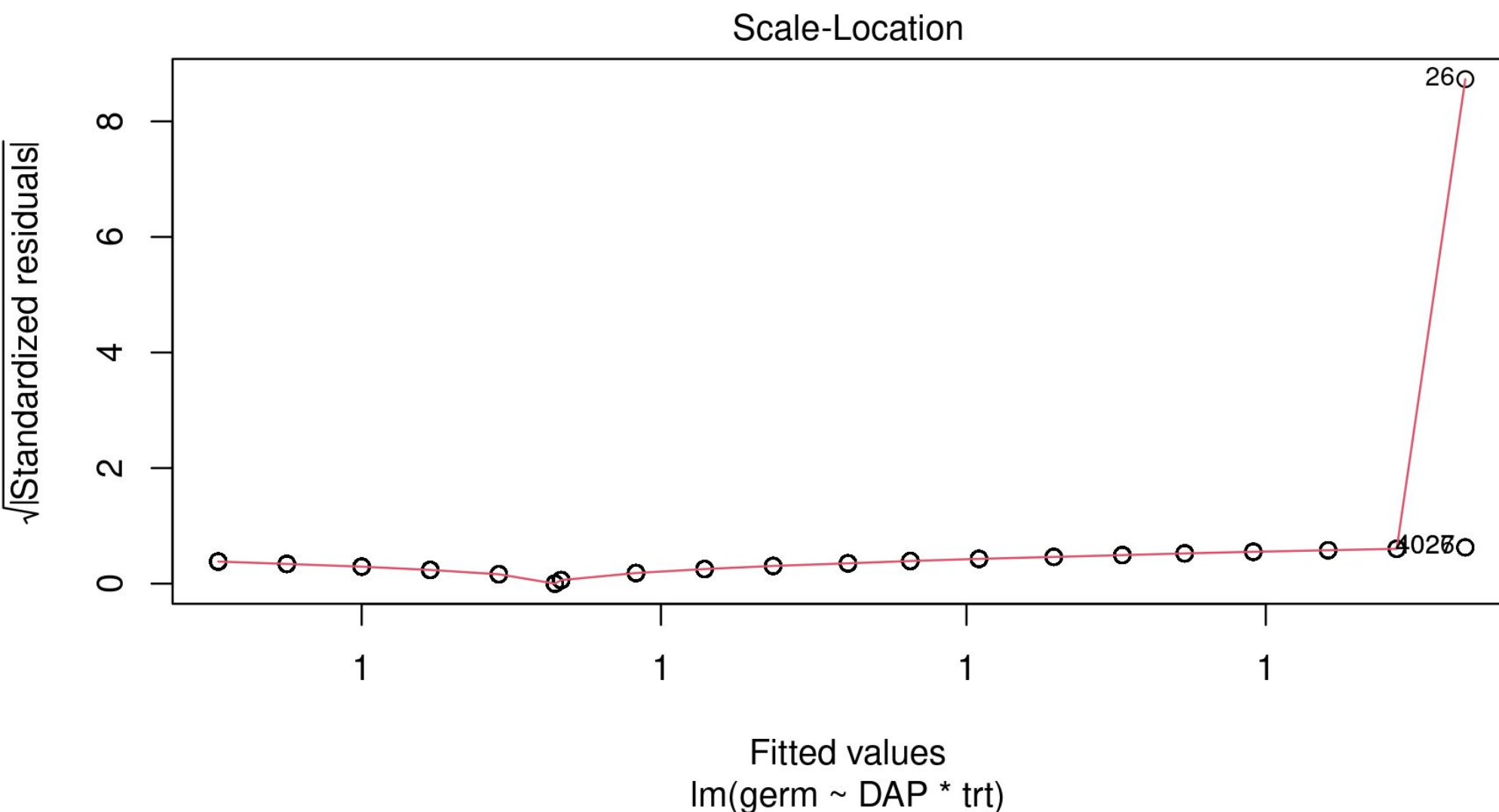


Scenario 1: Exercise 6 – no filters

```
> plot(m6)  
Hit <Return> to see next plot:  
Hit <Return> to see next plot:  
Hit <Return> to see next plot:  
Hit <Return> to see next plot:
```

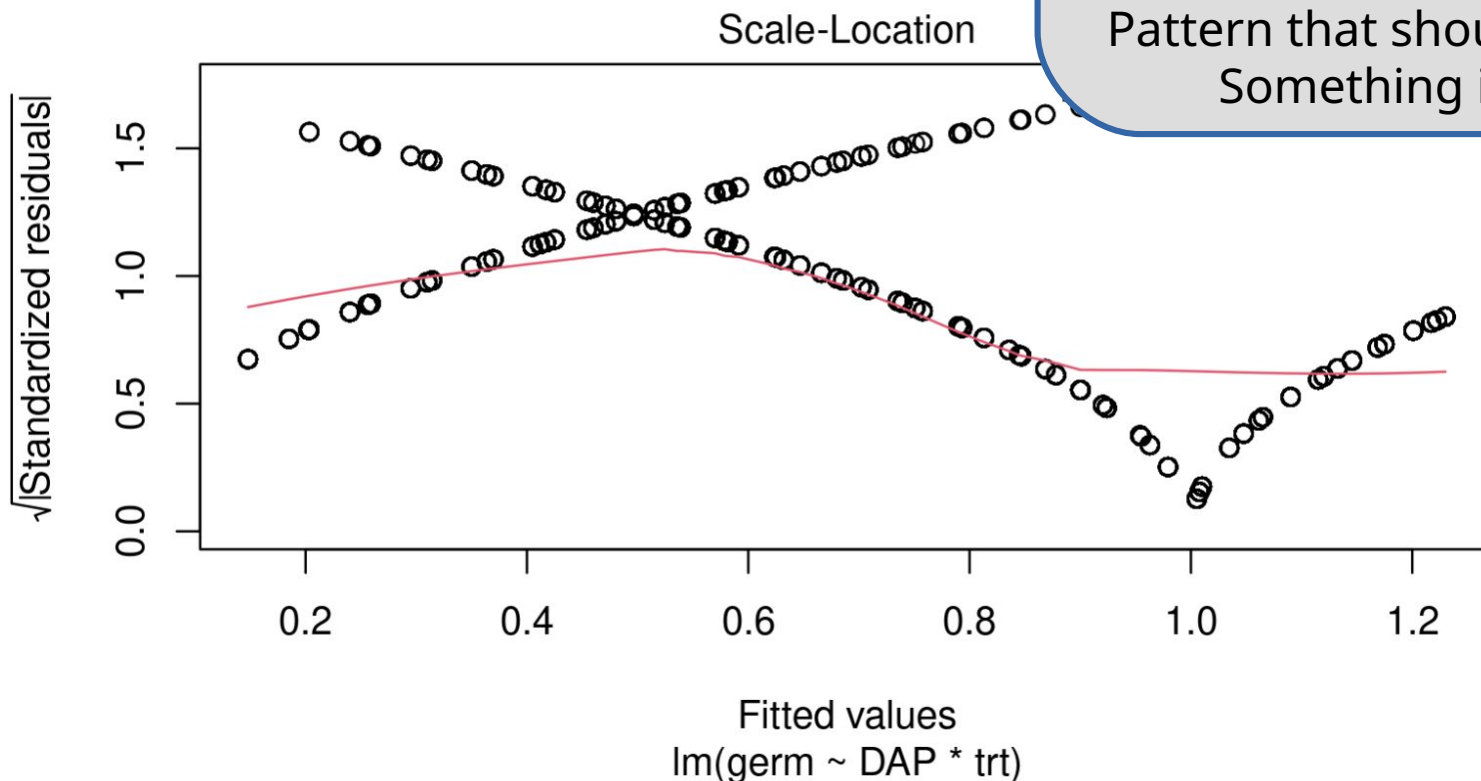
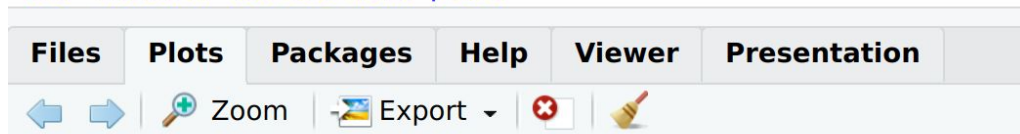


Scenario 1: Exercise 6 – with filters



Scenario 1: Exercise 6

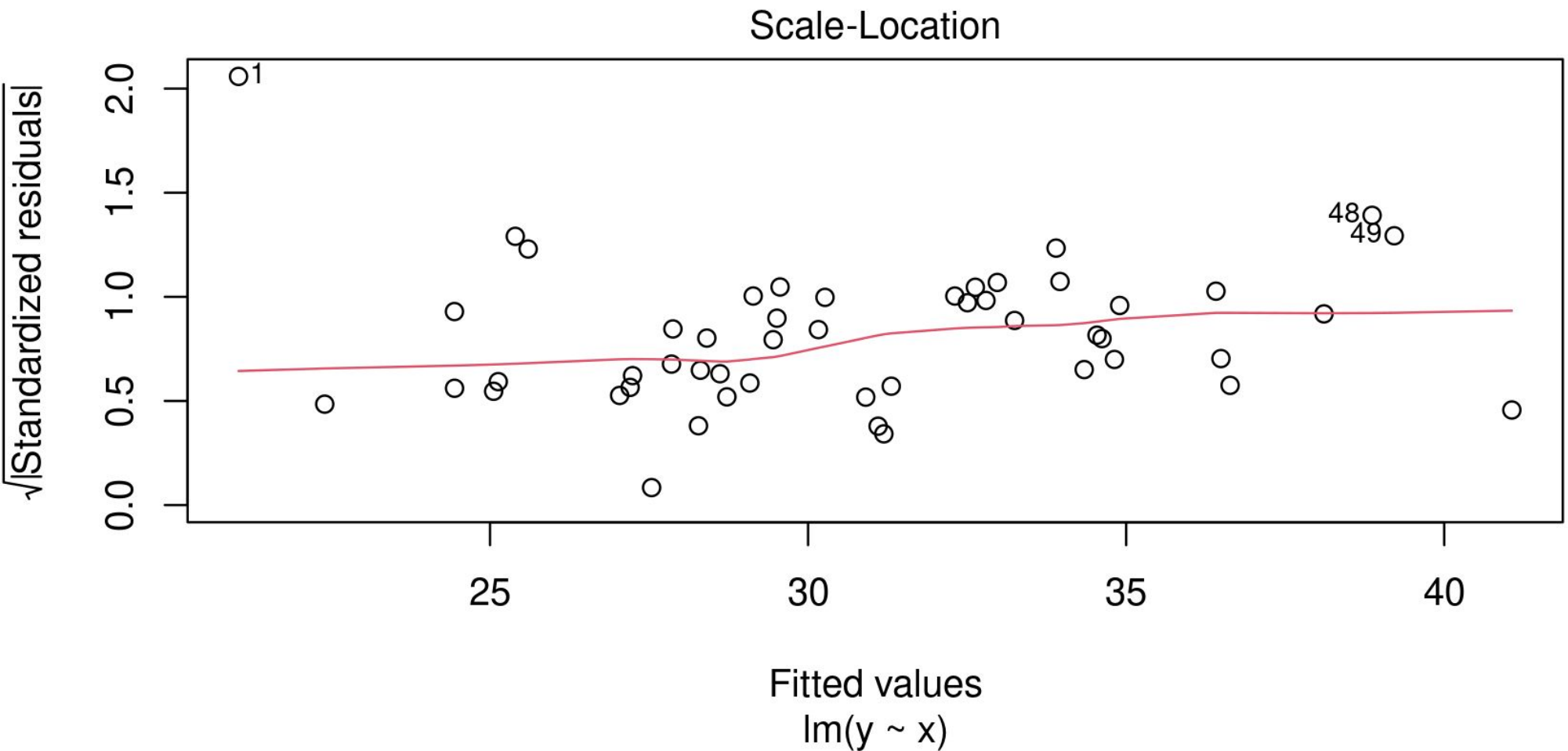
```
> plot(m6)  
Hit <Return> to see next plot:  
Hit <Return> to see next plot:  
Hit <Return> to see next plot:  
Hit <Return> to see next plot:
```



A scale-location plot can be Used to diagnose fit problems Such as heteroskedasticity. When Data are spread out on X a lot it Might be easier to read this than The normal residuals-fitted plot.

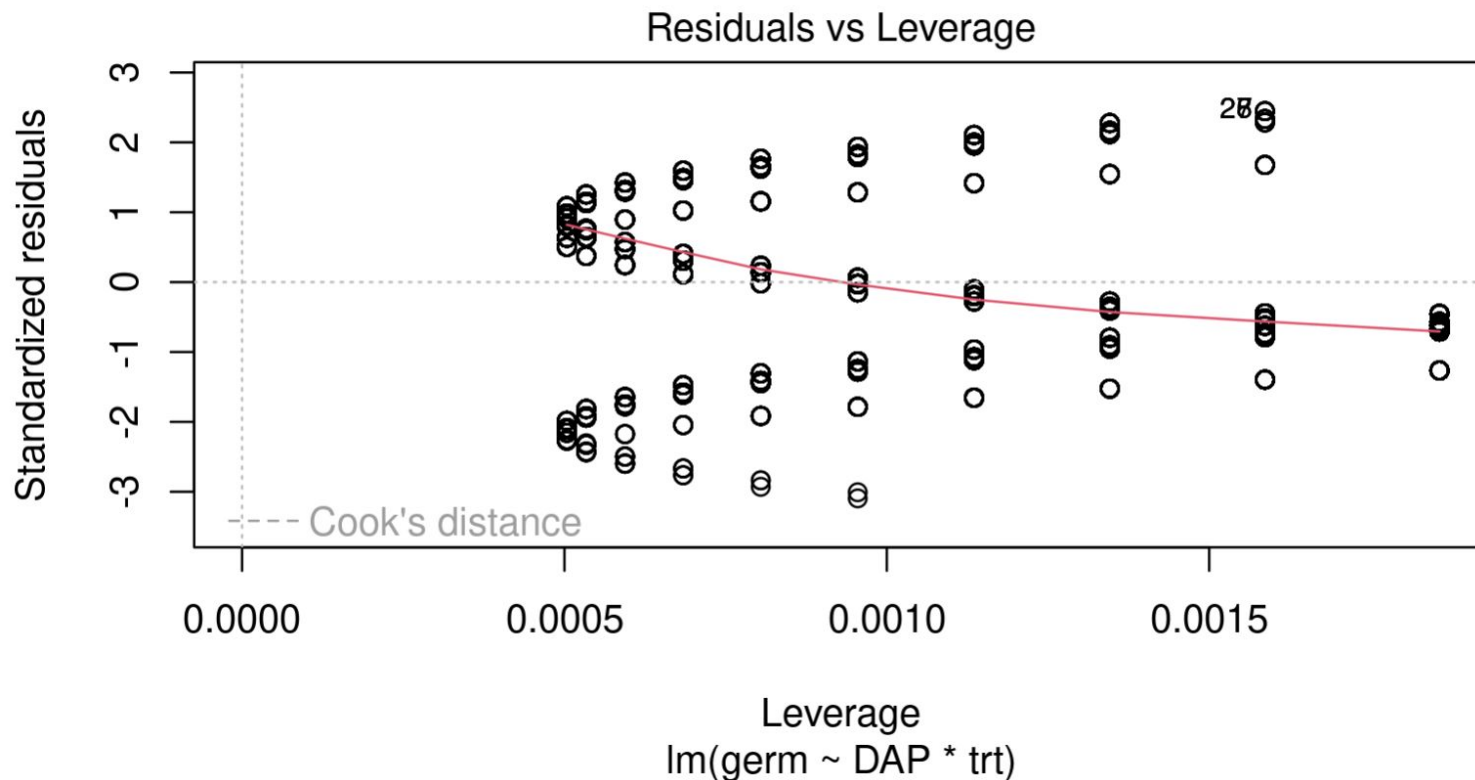
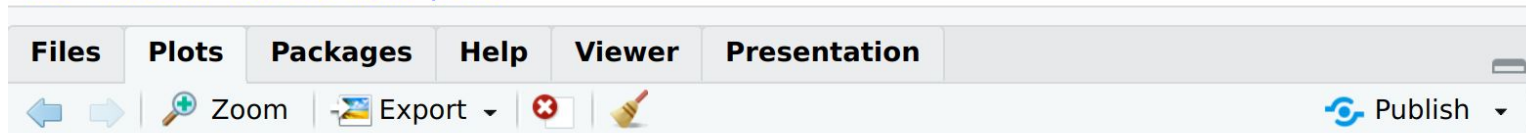
In this example we see a strange Pattern that should tell us that Something is wrong.

Scenario 1: Exercise 6 - Good

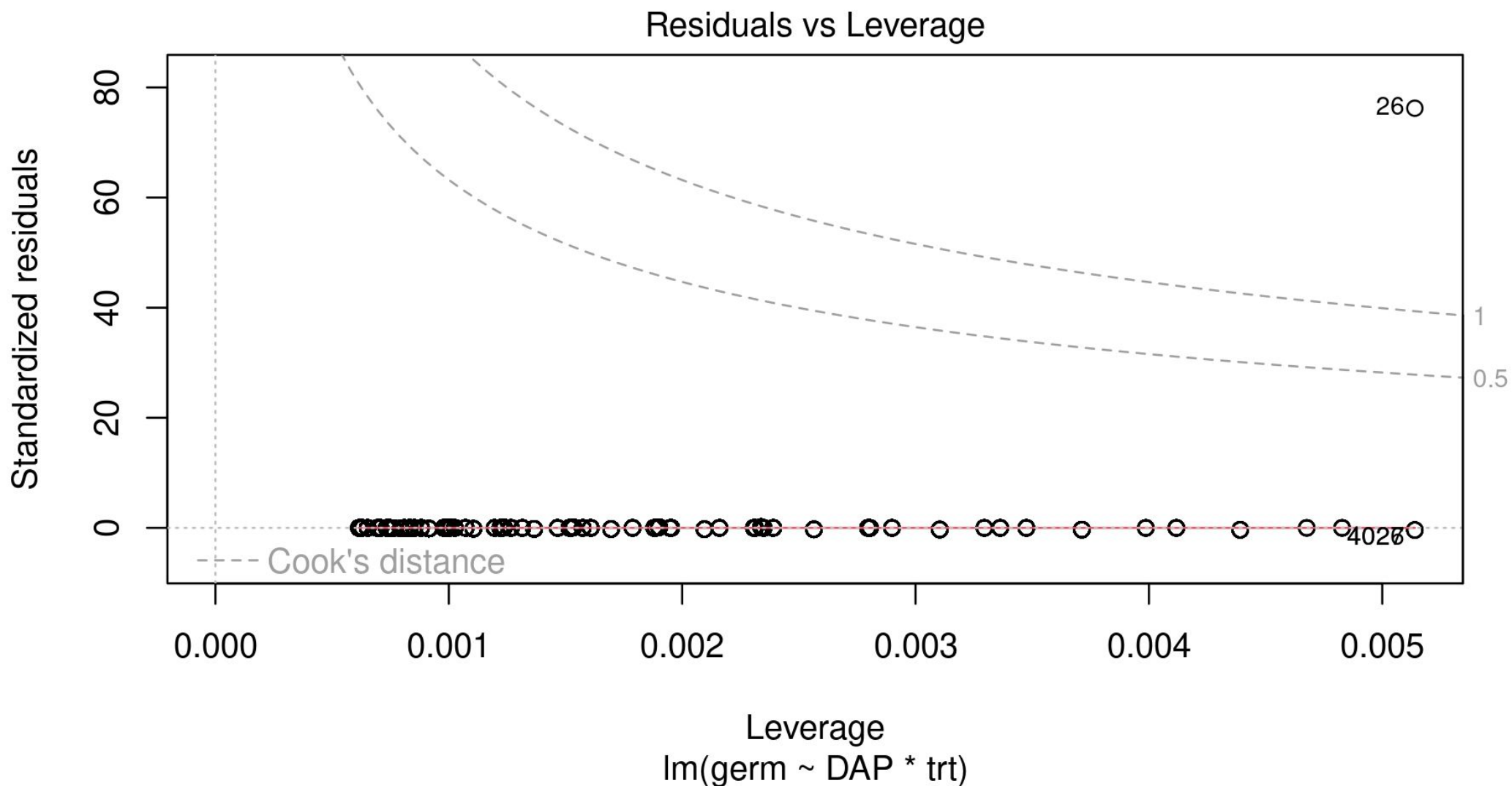


Scenario 1: Exercise 6 – no filters

```
> plot(m6)  
Hit <Return> to see next plot:  
Hit <Return> to see next plot:  
Hit <Return> to see next plot:  
Hit <Return> to see next plot:
```

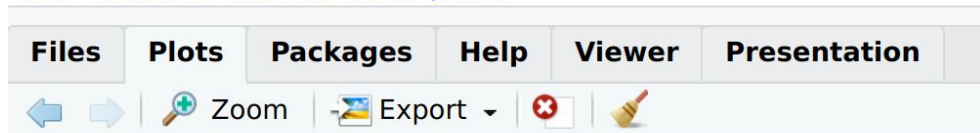


Scenario 1: Exercise 6 – with filters

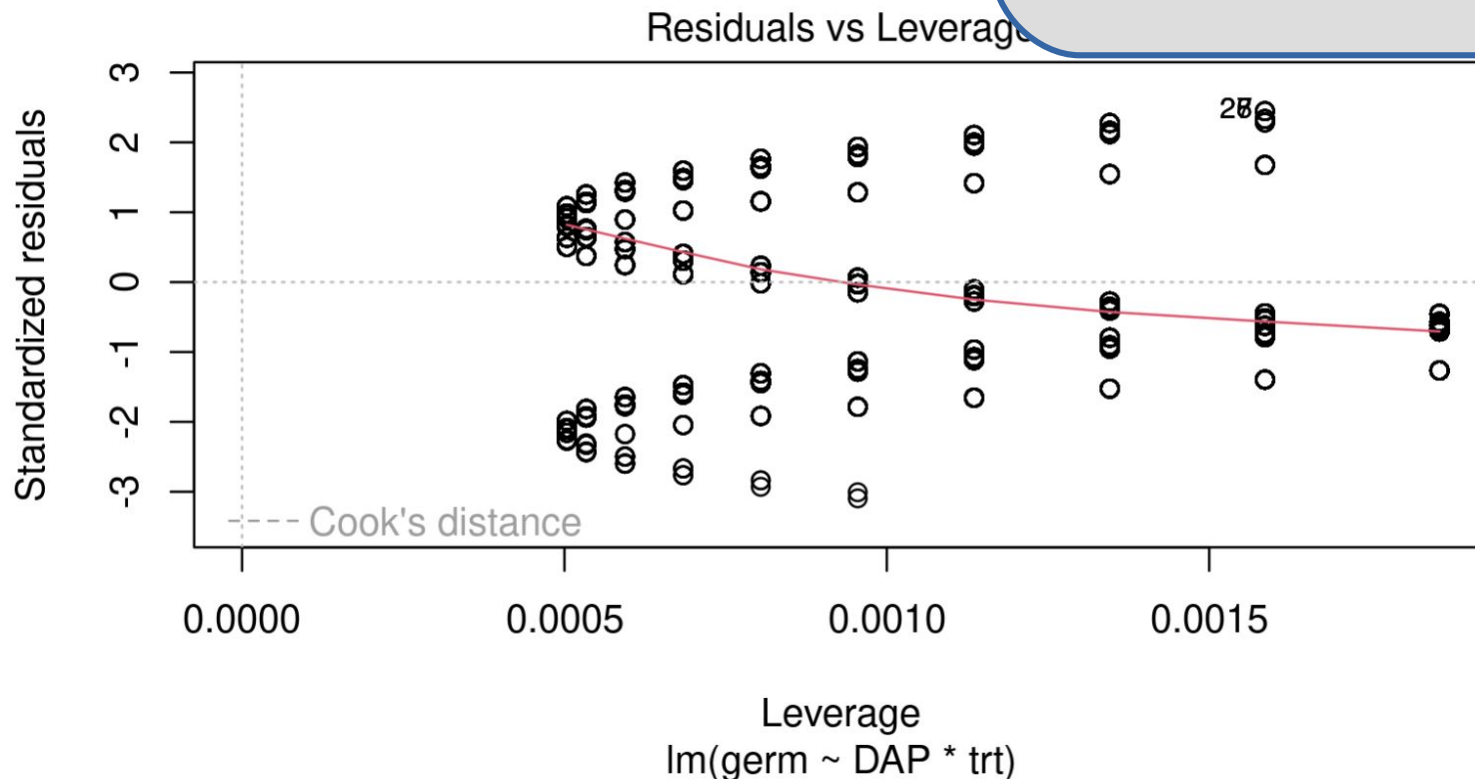


Scenario 1: Exercise 6

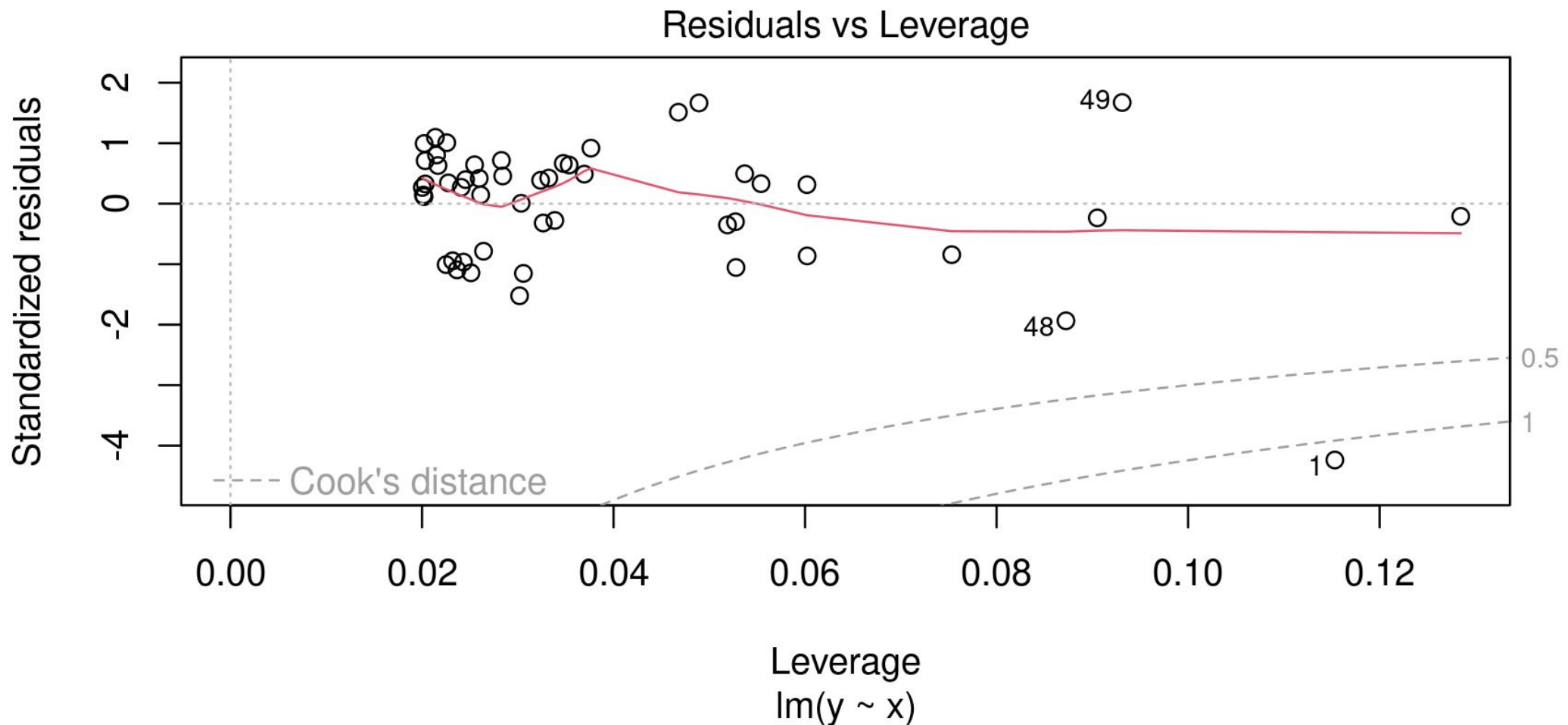
```
> plot(m6)  
Hit <Return> to see next plot:  
Hit <Return> to see next plot:  
Hit <Return> to see next plot:  
Hit <Return> to see next plot:
```



The residuals vs leverage plot
Shows us the influence of each
Data point.
We want to see data within the
Dashed Cook's Distance lines.
We do not see outliers based on
This, but the pattern is strange.



Scenario 1: Exercise 6 - GOOD



Scenario 1: Exercise 6



plant germination data analysis



All

Images

Videos

Forums

Shopping

More

Tools

Example

Pdf

Ppt

Scholarly articles for **plant germination data analysis**

Germination data analysis by time-to-event approaches - Romano - Cited by 27

Review of **data analysis** methods for seed **germination** ... - Scott - Cited by 965

Analysis of germination data from agricultural ... - Ritz - Cited by 148

Germination data are analyzed by several methods, which can be mainly classified as **germination indexes and traditional regression techniques to fit non-linear parametric functions to the temporal sequence of cumulative germination.** May 12, 2020



National Institutes of Health (NIH) (.gov)

<https://www.ncbi.nlm.nih.gov/articles/PMC7285257>

Germination Data Analysis by Time-to-Event Approaches - PMC

Scenario 1: Exercise 6

time to event data



Images

Example

Perspectives

Videos

Kaplan meier

Survival analysis

Repeated time

About 4,140,000,000 results (0.29 seconds)

Time-to-event (TTE) data is unique because the outcome of interest is not only whether or not an event occurred, but also when that event occurred. Traditional methods of logistic and linear regression are not suited to be able to include both the event and time aspects as the outcome in the model.






columbia.edu

<https://www.publichealth.columbia.edu> > research > time... ⋮


Time-To-Event (TTE) Data Analysis | Columbia Public Health

Scenario 1: Exercise 6

×   


[Example](#) [Images](#) [Perspectives](#) [Free](#) [Videos](#) [News](#) [Shopping](#) [Books](#) [Maps](#)

About 2,410,000,000 results (0.37 seconds)

 [r-project.org](https://cran.r-project.org)
[https://cran.r-project.org > packages > finalfit > vignettes](https://cran.r-project.org/packages/finalfit/vignettes) ⋮

Time-to-event (Survival) - CRAN

function. It produces a table containing counts (proportions) for factors, mean (SD) for continuous **variables** and a univariable and multivariable CPH ...

 [emilyzabor.com](https://www.emilyzabor.com)
[https://www.emilyzabor.com > tutorials > survival_ana...](https://www.emilyzabor.com/tutorials/survival_ana...) ⋮

Survival Analysis in R - Emily Zabor

Jun 21, 2023 — Survival **data** are **time-to-event data** that consist of a distinct start time and end time. Examples from cancer: Time from surgery to death ...

Scenario 1: Exercise 6

```
Surv(lung$time, lung$status)[1:10]
```

```
## [1] 306 455 1010+ 210 883 1022+ 310 361 218 166
```

We see that subject 1 had an event at time 306 days, subject 2 had an event at time 455 days, subject 3 was censored at time 1010 days, etc.

The `survfit()` function creates survival curves using the Kaplan-Meier method based on a formula. Let's generate the overall survival curve for the entire cohort, assign it to object `s1`, and look at the structure using `str()`:

```
s1 <- survfit(Surv(time, status) ~ 1, data = lung)
str(s1)
```

Scenario 1: Exercise 6

```
> m6_1<-survreg(Surv(DAP, germ) ~ trt, data = e5)
> summary(m6_1)
```

Call:

```
survreg(formula = Surv(DAP, germ) ~ trt, data = e5)
```

	Value	Std. Error	z	p
(Intercept)	2.63723	0.00816	323.02	< 2e-16
trtt	0.03595	0.01174	3.06	0.00219
trtu	0.03929	0.01176	3.34	0.00084
trty	0.04867	0.01191	4.09	4.4e-05
Log(scale)	-1.12846	0.01085	-104.01	< 2e-16

Scale= 0.324

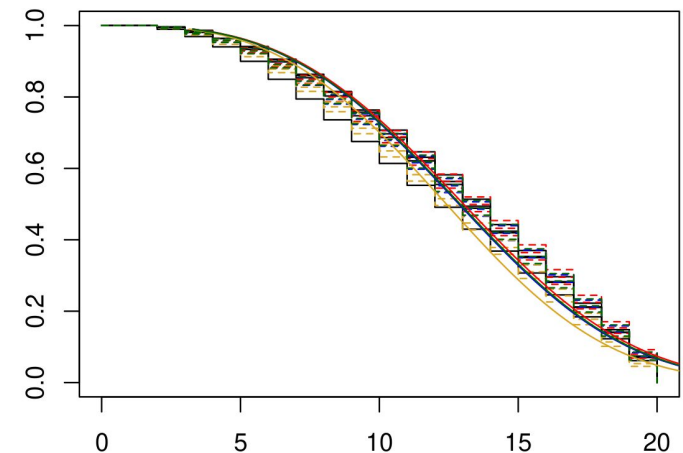
Weibull distribution

Loglik(model)= -17207.4 Loglik(intercept only)= -17217.3

Chisq= 19.82 on 3 degrees of freedom, p= 0.00018

Number of Newton-Raphson Iterations: 6

n= 5818



Scenario 1: Exercise 6

```
> m6_1<-survreg(Surv(DAP, germ) ~ trt, data = e5)
> summary(m6_1)
```

Call:

```
survreg(formula = Surv(DAP, germ) ~ trt, data = e5)
```

	Value
(Intercept)	2.63723
trtt	0.03595
trtu	0.03929
trty	0.04867
Log(scale)	-1.12846

Scale= 0.324

Weibull distribution

Loglik(model)= -17207.4 Loglik(intercept only)= -17217.3

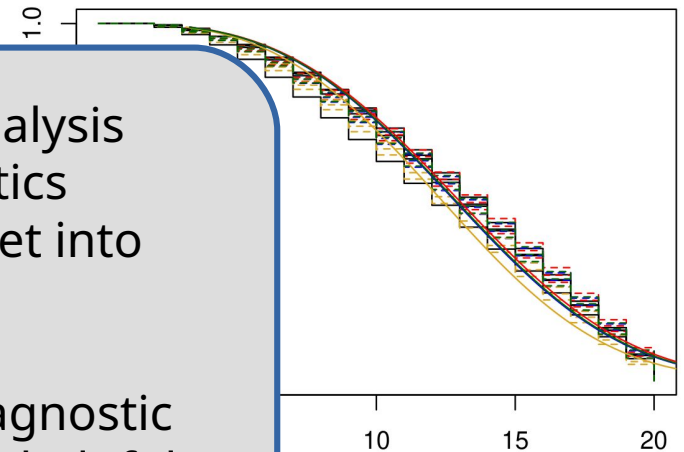
Chisq= 19.82 on 3 degrees of freedom, p= 0.00018

Number of Newton-Raphson Iterations: 6

n= 5818

The Stats in R/pcvr and Power Analysis seminars focus more on statistics so for the sake of time we won't get into Survival analysis today.

As general advice, searching for diagnostic plot options or your situation will be helpful.



Summary and Conclusion

- We've gone over several kinds of errors and why they appear.
- Introduced different packages for common tasks (tidyverse and data.table) for cases where you may not be comfortable with base R.
- Familiarized with general troubleshooting steps for R/similar languages.

Summary and Conclusion

- Read the error (and the docs)
- Isolate the problem area
- Simplify and make verbose
- Edit
- Test

Summary and Conclusion

- Finally, this is a new presentation and a new type of workshop for us. If you have feedback about how to improve this/what went well vs poorly for you then I'd love to know that.