

Working Habits

How to make your life easier and more efficient.

`b.rowlingson@gmail.com`
`@geospacedman`

*Faculty of Health and Medicine,
Lancaster University*

```
> readmydata=function(f, ...) {  
  d = read.csv(f, ...)  
  d$old=d$age>5;  
  d  
}  
> data = readmydata("data.csv")  
> fit = glm(y~x,data=data)  
> summary(fit)
```

Cut and Paste

script.R

```
# my first analysis
```

```
d = read.csv("data.csv", ...)  
d$old=d$age>5;  
fit = glm(y~x,data=d)  
summary(fit)
```

```
# another analysis  
d2 = read.csv("data2.csv")  
fit2 = glm(y~x,data=d2)  
summary(fit2)
```

```
# another analysis  
d2 = read.csv("data2.csv")  
fit2 = glm(y~x,data=d2)  
summary(fit2)
```

first.R

```
# my first analysis  
d = read.csv("data.csv", ...)  
d$old=d$age>5;  
fit = glm(y~x,data=d)  
summary(fit)
```

```
>source("first.R")
```



second.R

```
# another analysis  
d2 = read.csv("data2.csv")  
fit2 = glm(y~x,data=d2)  
summary(fit2)
```

(One) function per file



first.R

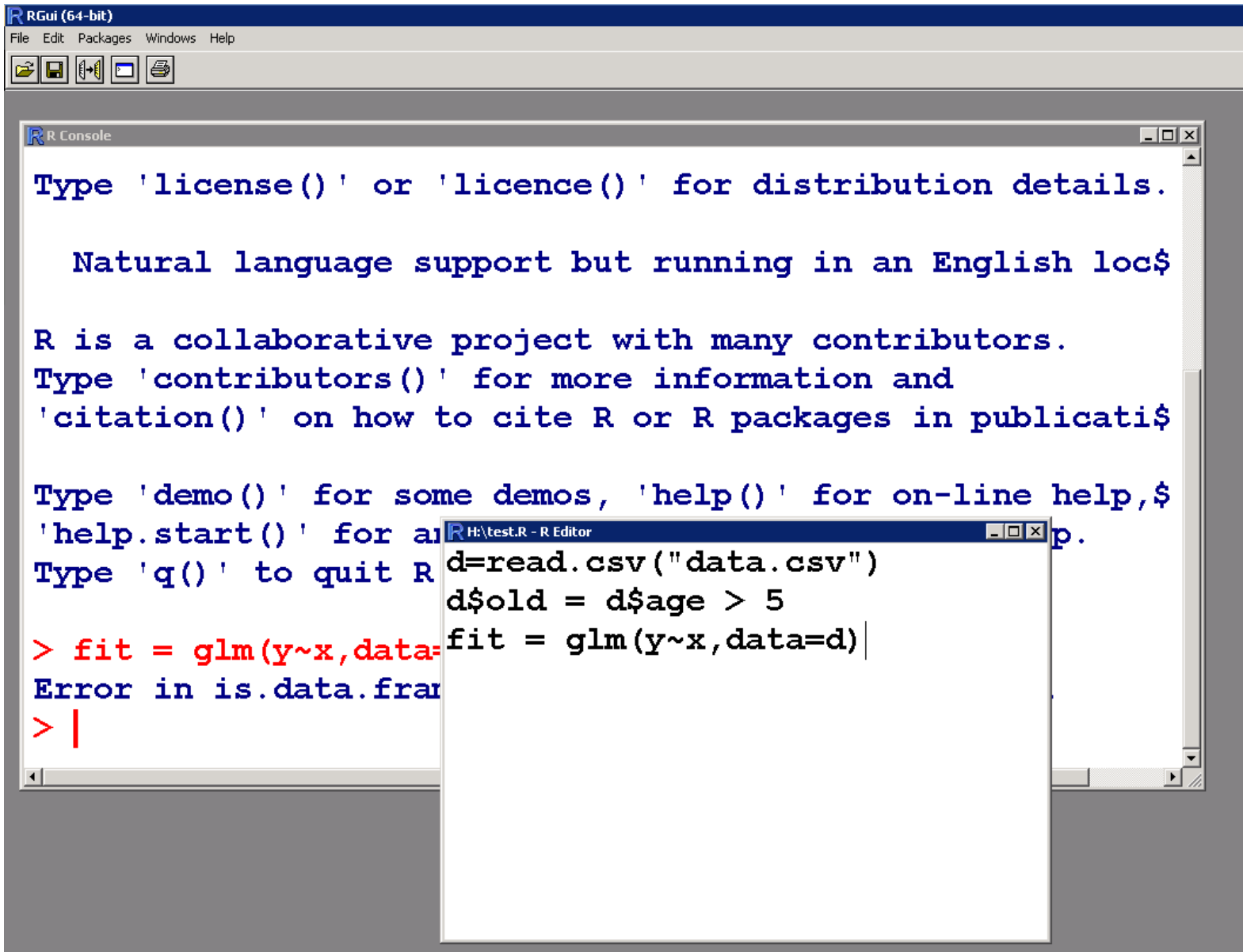
```
# my first analysis
first=function(f){
d = read.csv(f,...)
d$old=d$age>5;
fit = glm(y~x,data=d)
summary(fit)
}
```

```
> source('first.R')
> first('data.csv')
> source('second.R')
> second('data.csv')
> second('data2.csv')
```

second.R

```
# another analysis
second=function(f){
d2 = read.csv(f)
fit2 = glm(y~x,data=d2)
summary(fit2)
}
```

Windows GUI



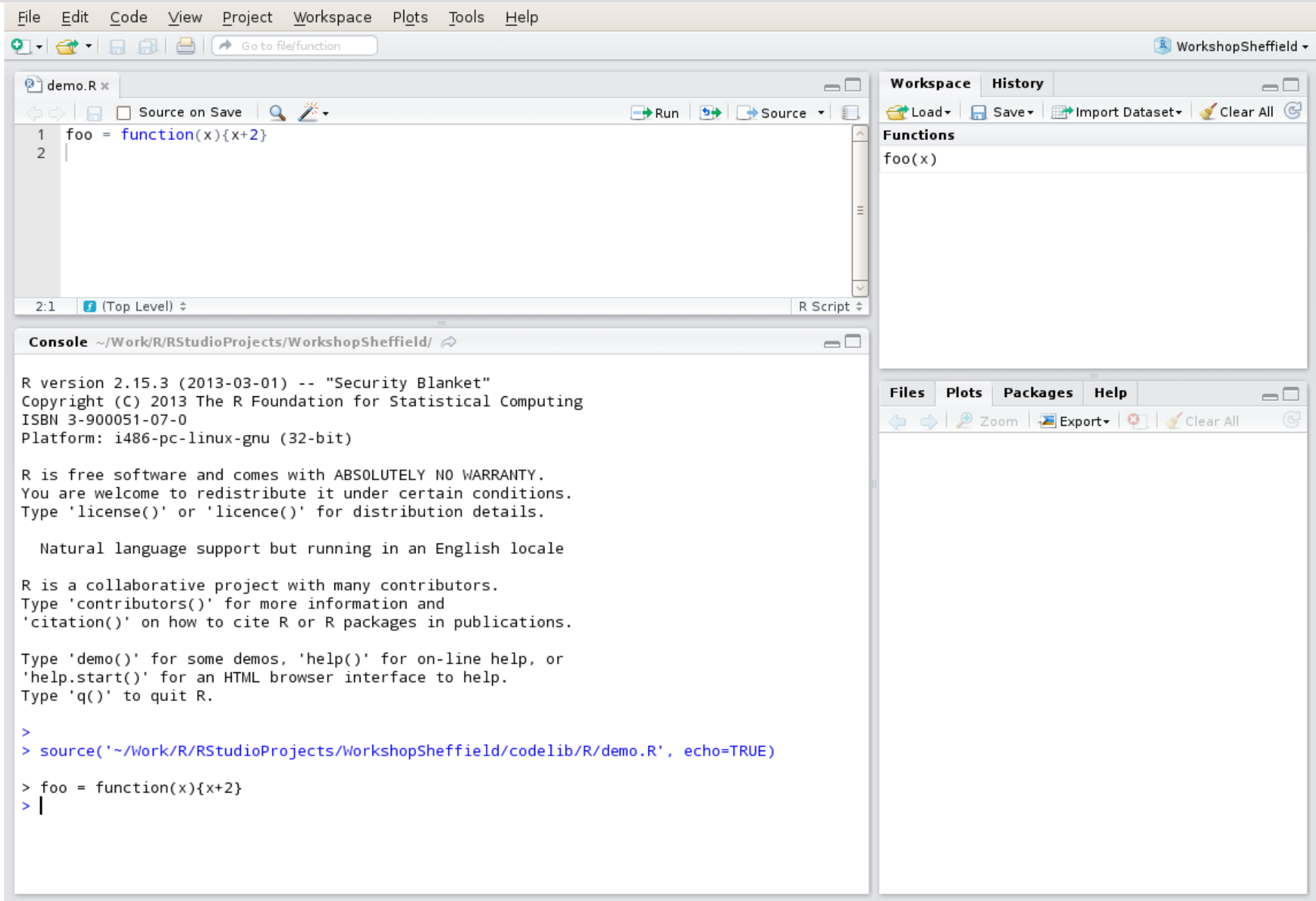
The screenshot shows the RGui (64-bit) interface. The main window is the R Console, which displays the following text:

```
Type 'license()' or 'licence()' for distribution details.  
  
Natural language support but running in an English locale  
  
R is a collaborative project with many contributors.  
Type 'contributors()' for more information and  
'citation()' on how to cite R or R packages in publications.  
  
Type 'demo()' for some demos, 'help()' for on-line help, or  
'help.start()' for an HTML browser interface to help.  
Type 'q()' to quit R.  
  
> fit = glm(y~x,data=d)  
Error in is.data.frame(x) :  
  argument is not a data frame  
> |
```

An R Editor window is also open, showing the following code:

```
H:\test.R - R Editor  
d=read.csv("data.csv")  
d$old = d$age > 5  
fit = glm(y~x,data=d)|
```

RStudio



The screenshot displays the RStudio environment with the following components:

- Source Editor:** Contains a script named `demo.R` with the following code:

```
1 foo = function(x){x+2}
2 |
```
- Console:** Shows the R version information and the execution of the script:

```
R version 2.15.3 (2013-03-01) -- "Security Blanket"
Copyright (C) 2013 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: i486-pc-linux-gnu (32-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

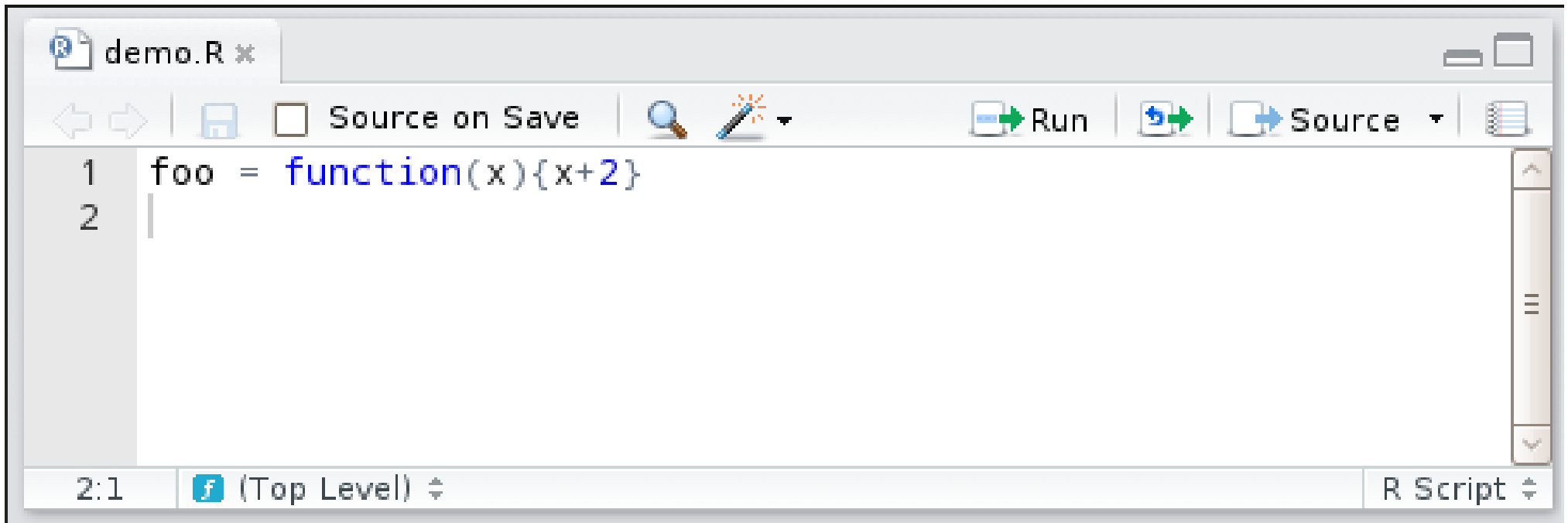
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
> source('~/.Work/R/RStudioProjects/WorkshopSheffield/codelib/R/demo.R', echo=TRUE)

> foo = function(x){x+2}
> |
```
- Workspace:** Shows the function `foo(x)` loaded into the environment.
- Files:** Shows the current project structure.

RStudio

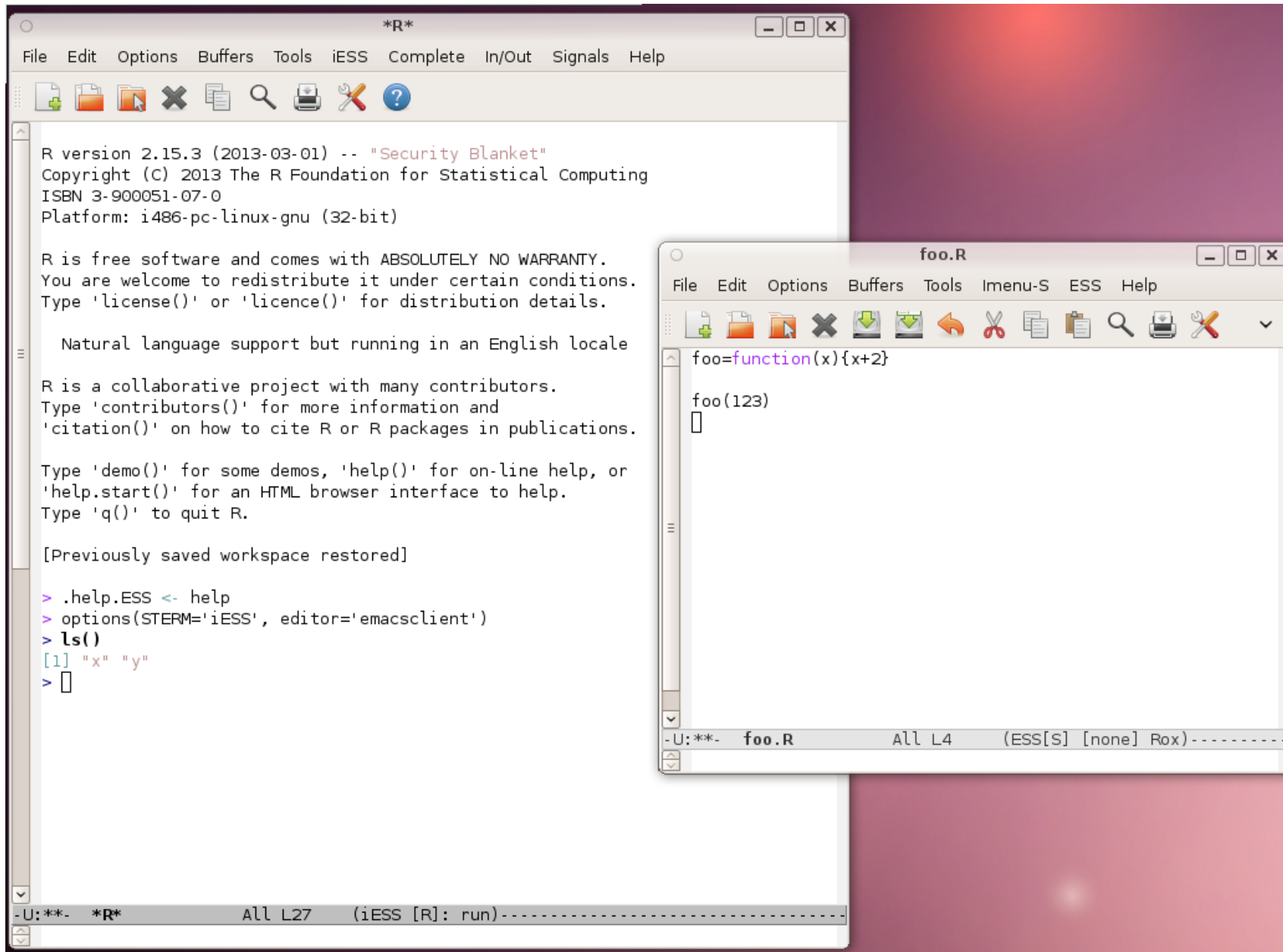


The screenshot shows the RStudio editor window for a file named 'demo.R'. The editor contains two lines of R code: a function definition on line 1 and a blank line on line 2. The function is named 'foo' and takes a parameter 'x', returning 'x+2'. The RStudio interface includes a toolbar with icons for navigation, saving, and running code, as well as a status bar at the bottom indicating the current cursor position and the active environment.

```
1 foo = function(x){x+2}  
2 |
```

2:1 f (Top Level) R Script

Emacs/ESS



The screenshot displays the Emacs/ESS environment. The main window, titled `*R*`, shows the R version 2.15.3 (2013-03-01) running on a 32-bit Linux platform. It includes the R license text and a list of commands: `.help.ESS`, `options(STERM='iESS', editor='emacsclient')`, and `ls()`. The `ls()` command output is `[1] "x" "y"`. The status bar at the bottom indicates the current buffer is `*R*` and the mode is `(iESS [R]: run)`.

A secondary window, titled `foo.R`, shows a custom R script. The script defines a function `foo` that takes an argument `x` and returns `x+2`. The function is then called with `foo(123)`, resulting in an empty output buffer. The status bar for this window shows the buffer is `foo.R` and the mode is `(ESS[S] [none] Rox)`.

Source problem

- Everything is in **1s** ()
- Need to **save** everything
- Or **source** everything
- Stuff copied everywhere
- Solution: ***organise things nicely***

Package devtools

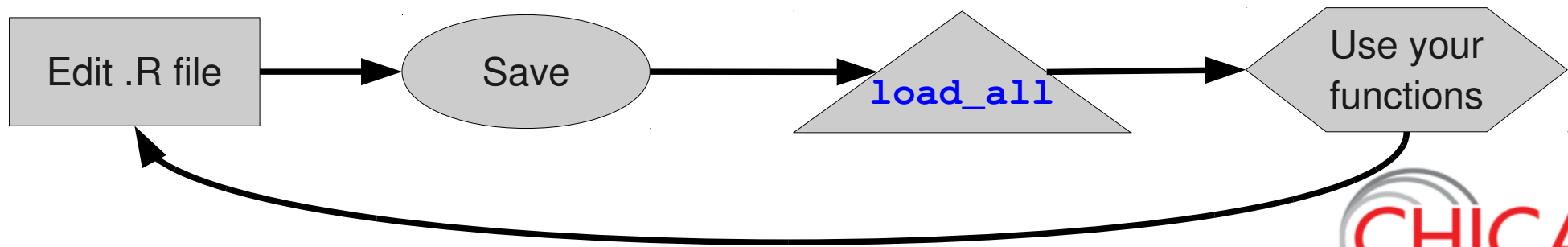
Be functional.

Write functions not scripts

```
> require(devtools)
> create('Code/mycode')
```

Now edit files in `Code/mycode/R/whatever.R`

```
> load_all('Code/mycode')
```



Why bother?

```
> load_all("Code/mycode")
Loading mycode
Adding files missing in collate: test.R
```

Load my code

```
> z = foo(1,2,3)
```

Here's my new function

```
> ls()
[1] 'z'
```

Where's foo?

```
> search()
[1] ".GlobalEnv" "package:mycode" "package:roxygen2"
[4] "package:digest" "package:devtools" "package:stats"
[7] "package:graph" "package:grDevices" "package:utils"
[10] "package:data" "package:methods" "Autoloads"
[13] "package:base"
```

```
> ls(pos=2)
[1] "foo"
```

There it is

Package devtools

You've just written an R package

Handy conventions

▼ Project

▼ Code

▼ mymethod

▼ R

fitmodel.R

▼ projectcode

▼ R

fixupdata.R

readdata.R

runanalysis.R

Folder for your project

Folder for your R code

▼ Data

▼ 2013-06-22

dataset.csv

▼ 2013-06-28

dataset.csv

Folder for your datasets

Handy conventions

▼ Project

▼ Code

▼ mymethod

▼ R

fitmodel.R

Code that might have wide use

▼ projectcode

▼ R

fixupdata.R

readdata.R

runanalysis.R

Code only useful in this project

▼ Data

▼ 2013-06-22

dataset.csv

▼ 2013-06-28

dataset.csv

Create a new folder when you get updated datasets. Keep 'readme' files

Spin your scripts

```
#'  
#' This is my project  
#'  
#' First I create some data  
  
#+ create  
d = data.frame(x=runif(10), y=runif(10))  
  
#' now we do something else  
  
#+ testsum  
sum(d$x) # this is a number  
  
#+  
plot(d$x, d$y)  
  
#' we can now look at this plot  
  
> source('makeplot.R')  
> require(knitr); spin('makeplot.R')
```


Gmail

R Output

Home Page



file:///home/me/output/project.html



Search

This is my project

First I create some data

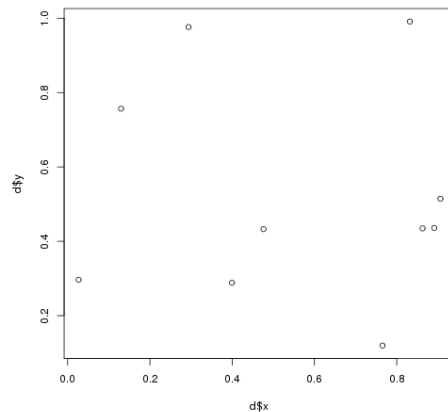
```
d = data.frame(x = runif(10), y = runif(10))
```

now we do something else

```
sum(d$x) # this is a number
```

```
## [1] 5.583
```

```
plot(d$x, d$y)
```



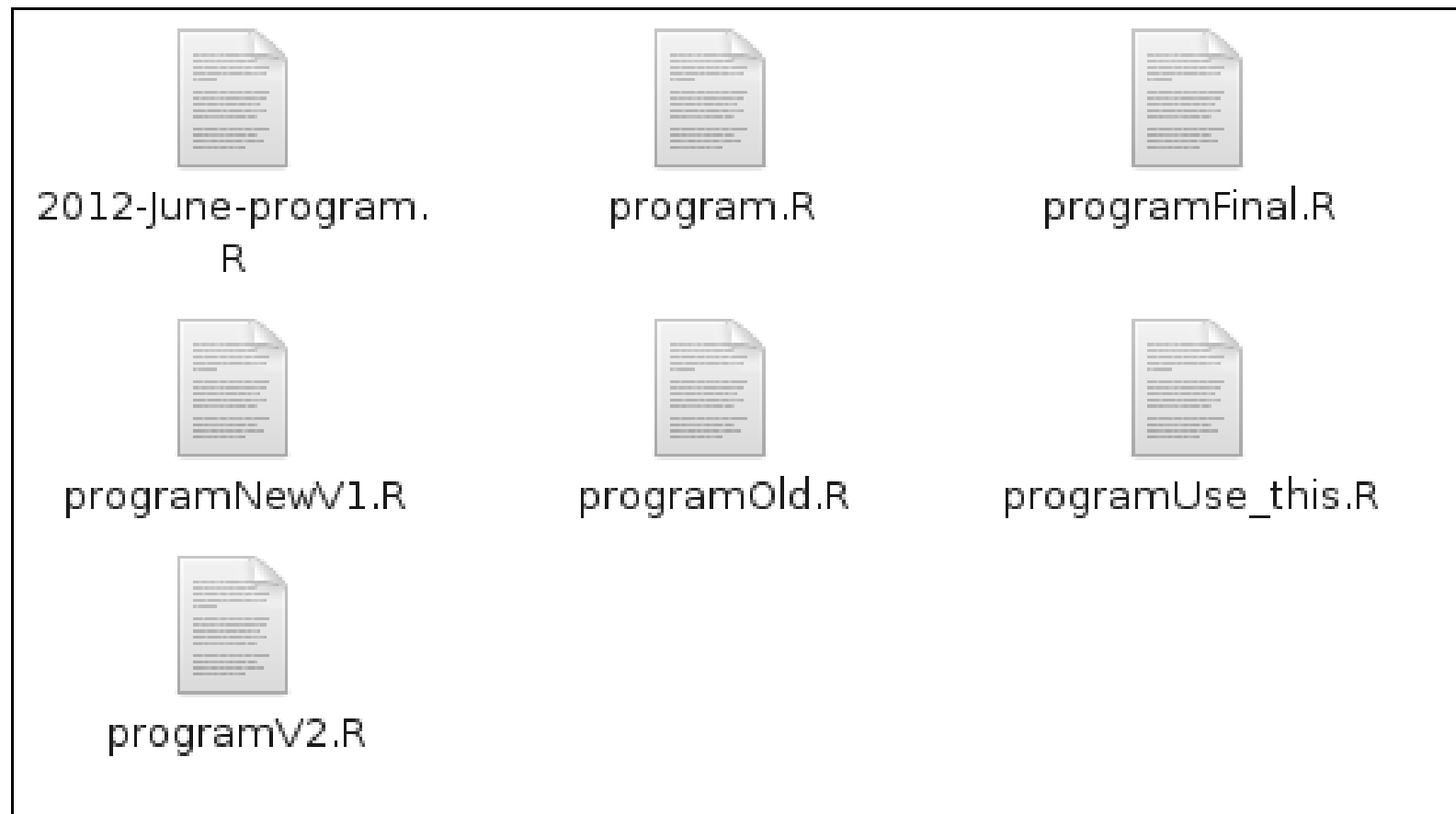
we can now look at this plot

The `knitr` package

- You put R code in a document
- Knitr processes the document
- Knitr embeds results and graphics in the output
- Can produce HTML, LaTeX, PDF, Markdown, OpenOffice, MS Word
- Works with other languages
- Great for routine reporting, data-driven documents
- Reproducible Research

Revision control

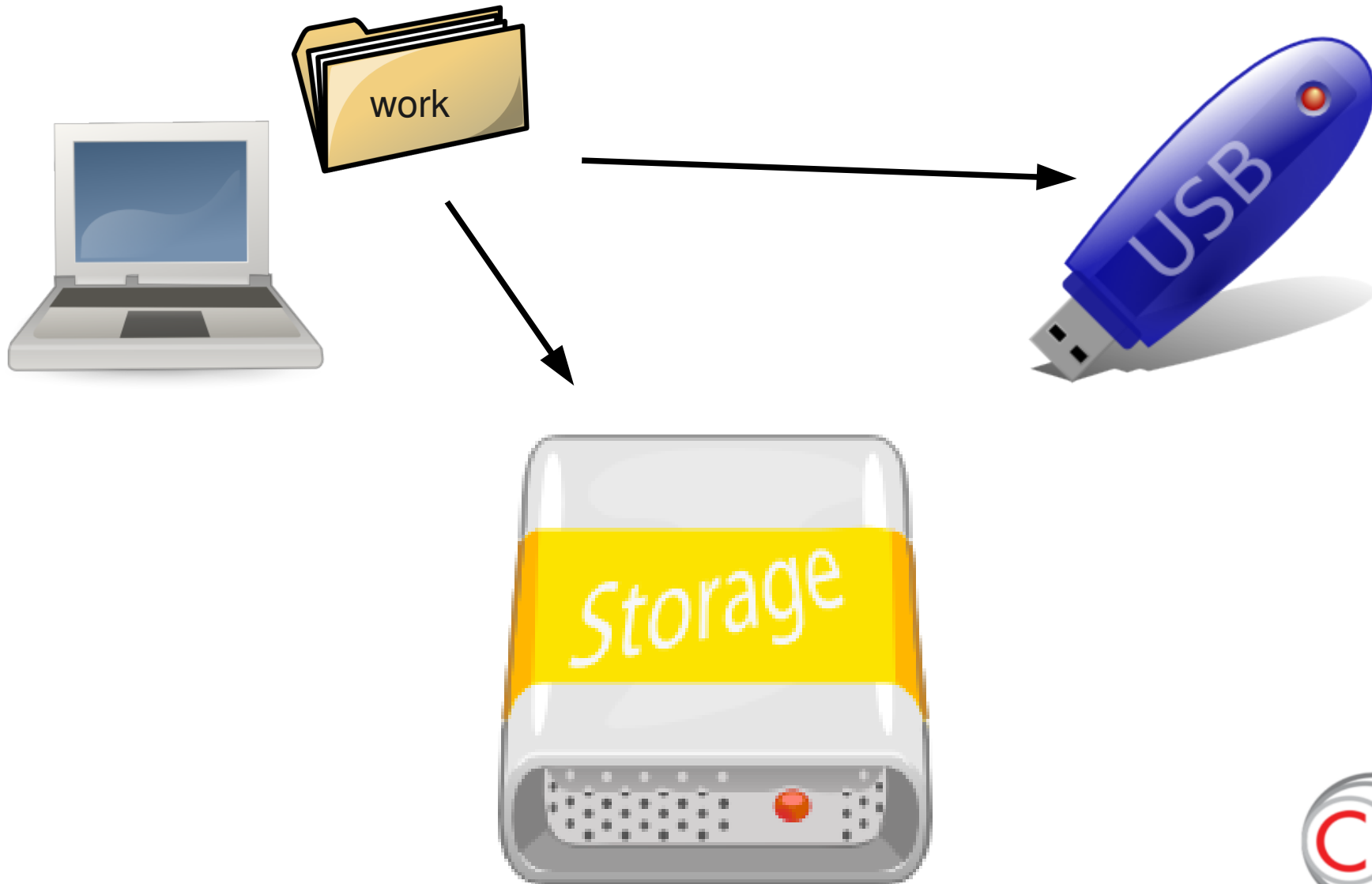
- Disk space is now cheap
- Why delete when you can keep?



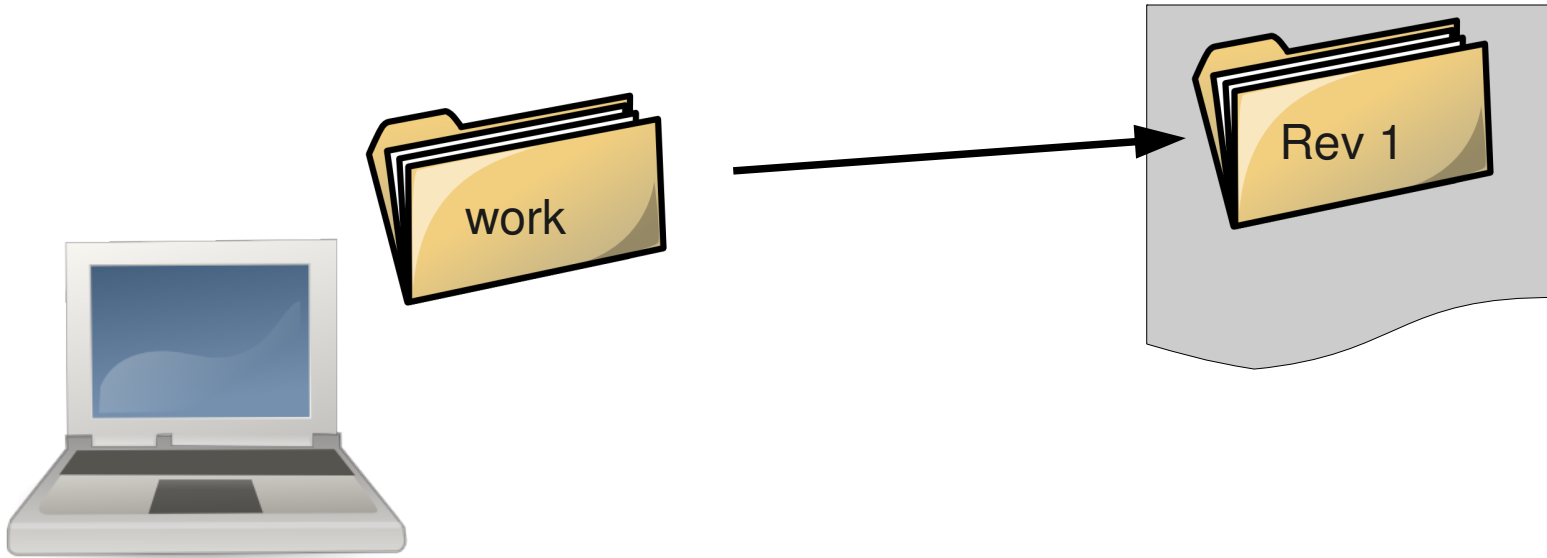
What you really want...

- What did this file look like
 - Last Week?
 - Last Year?
 - When I did the analysis for the paper?
- What's changed today?
- I want to try something, but it might not work out...
 - How can I go back?

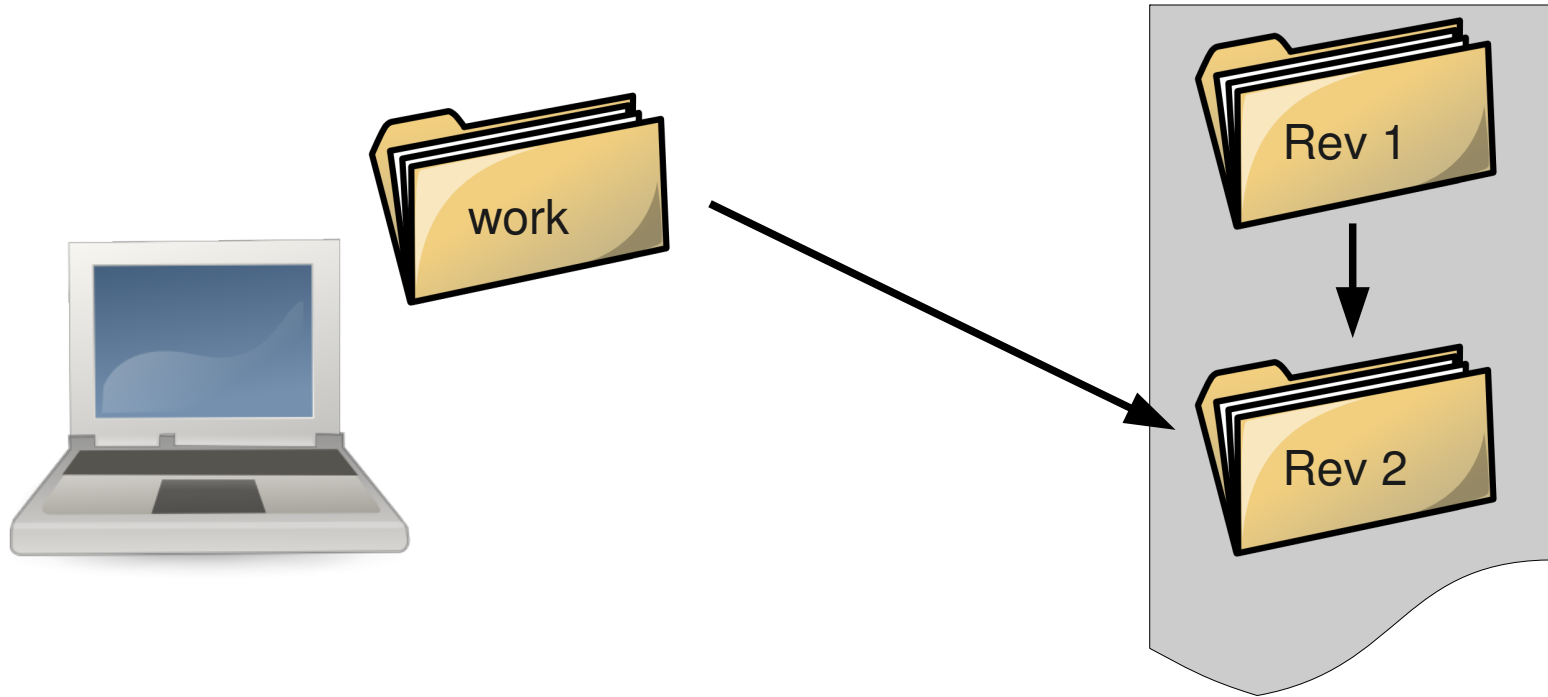
Is This Just Backup?



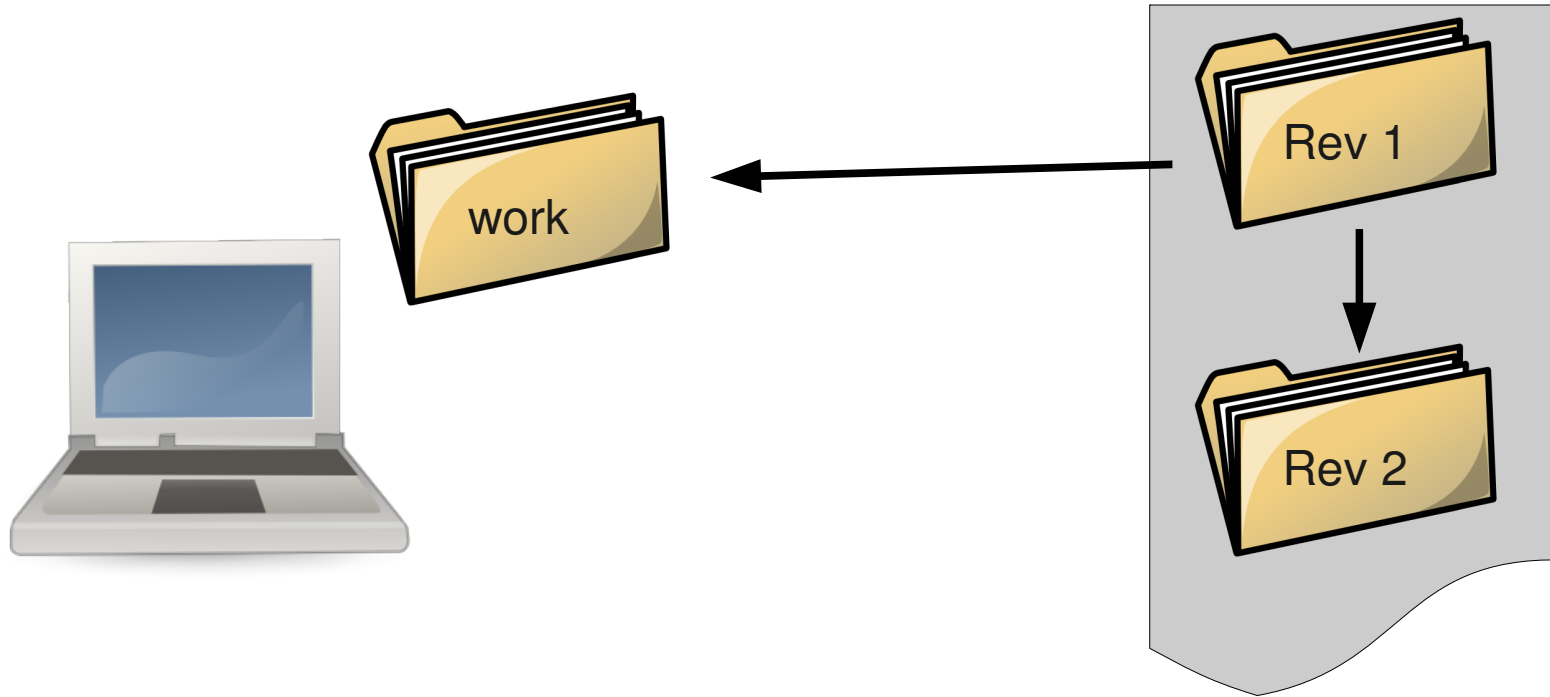
initialise



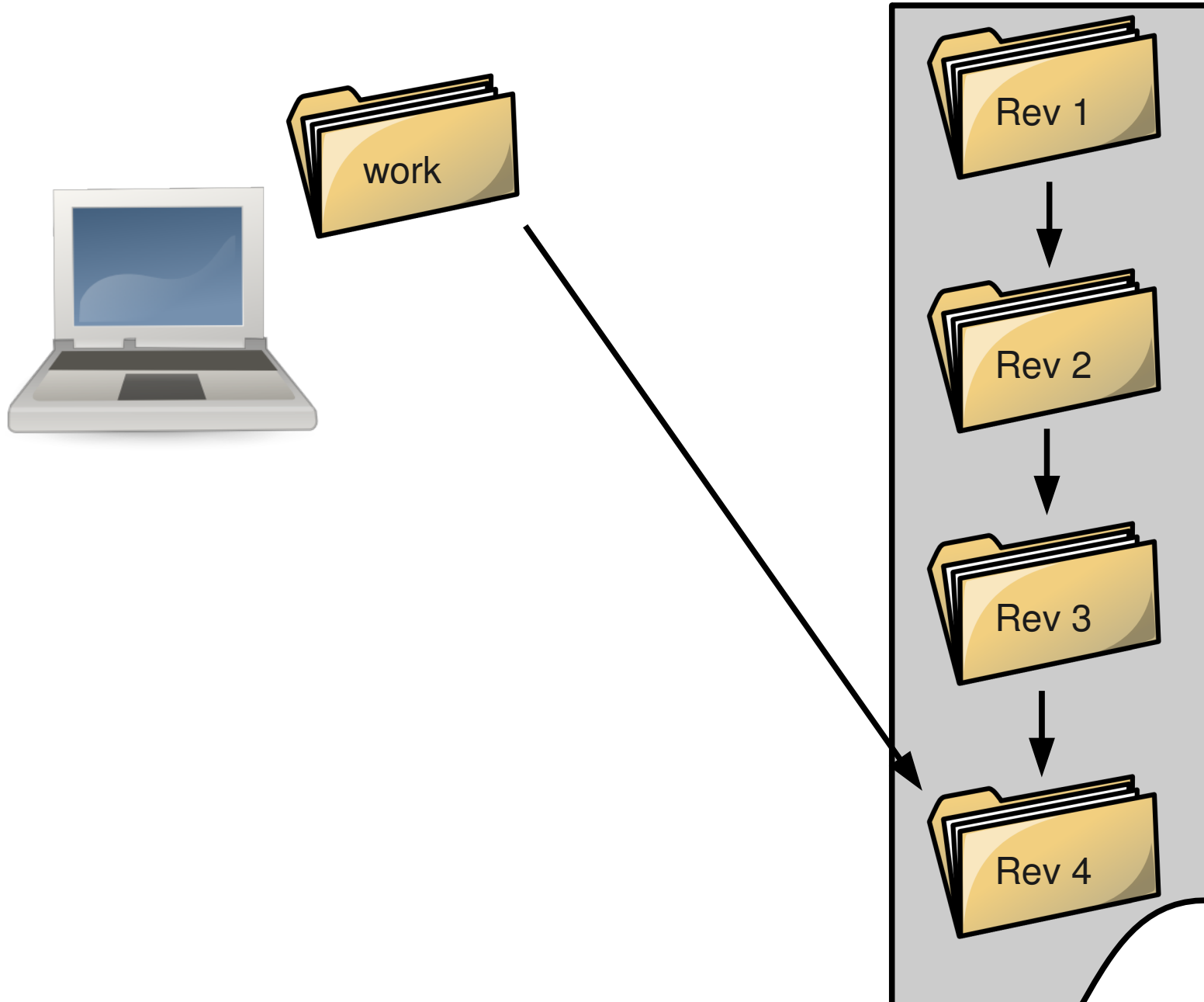
commit



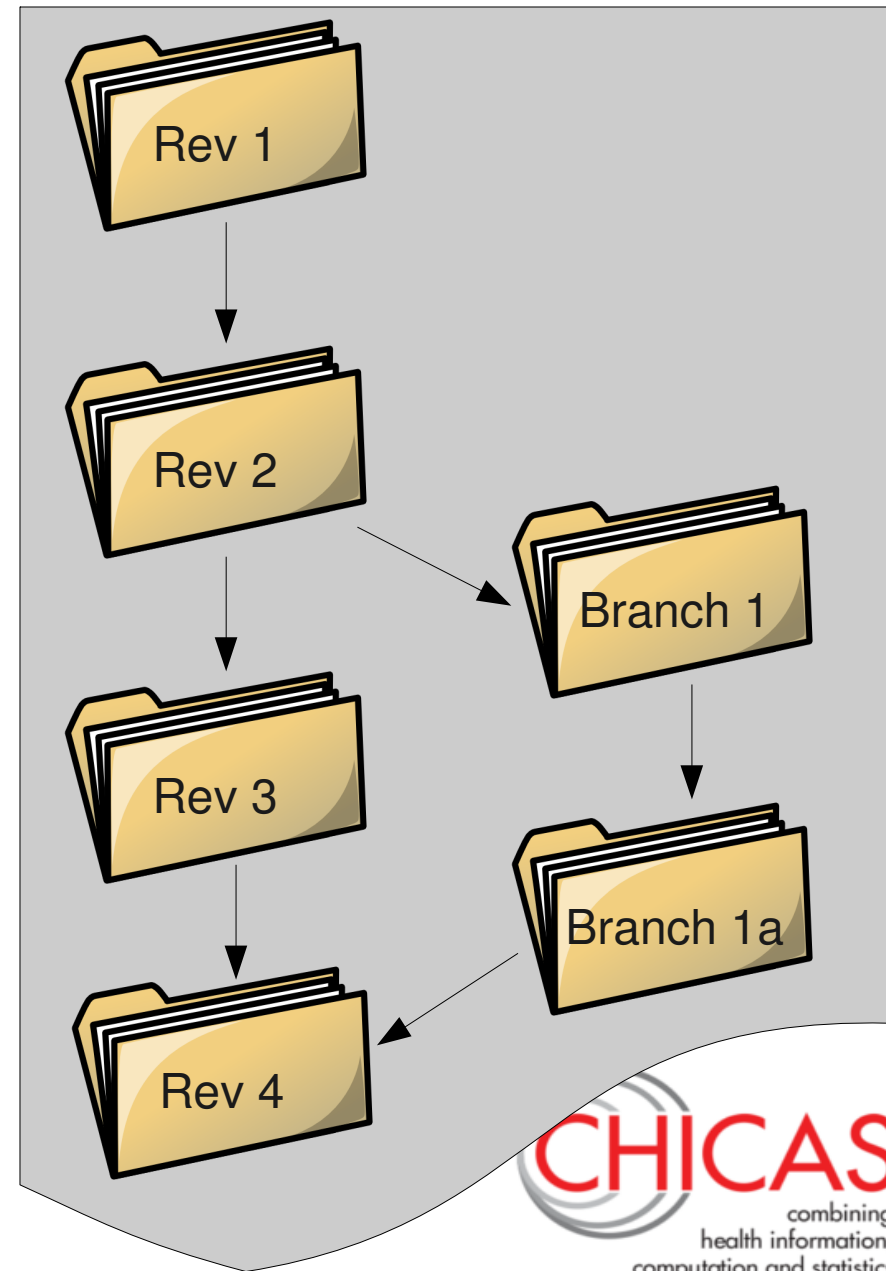
pull



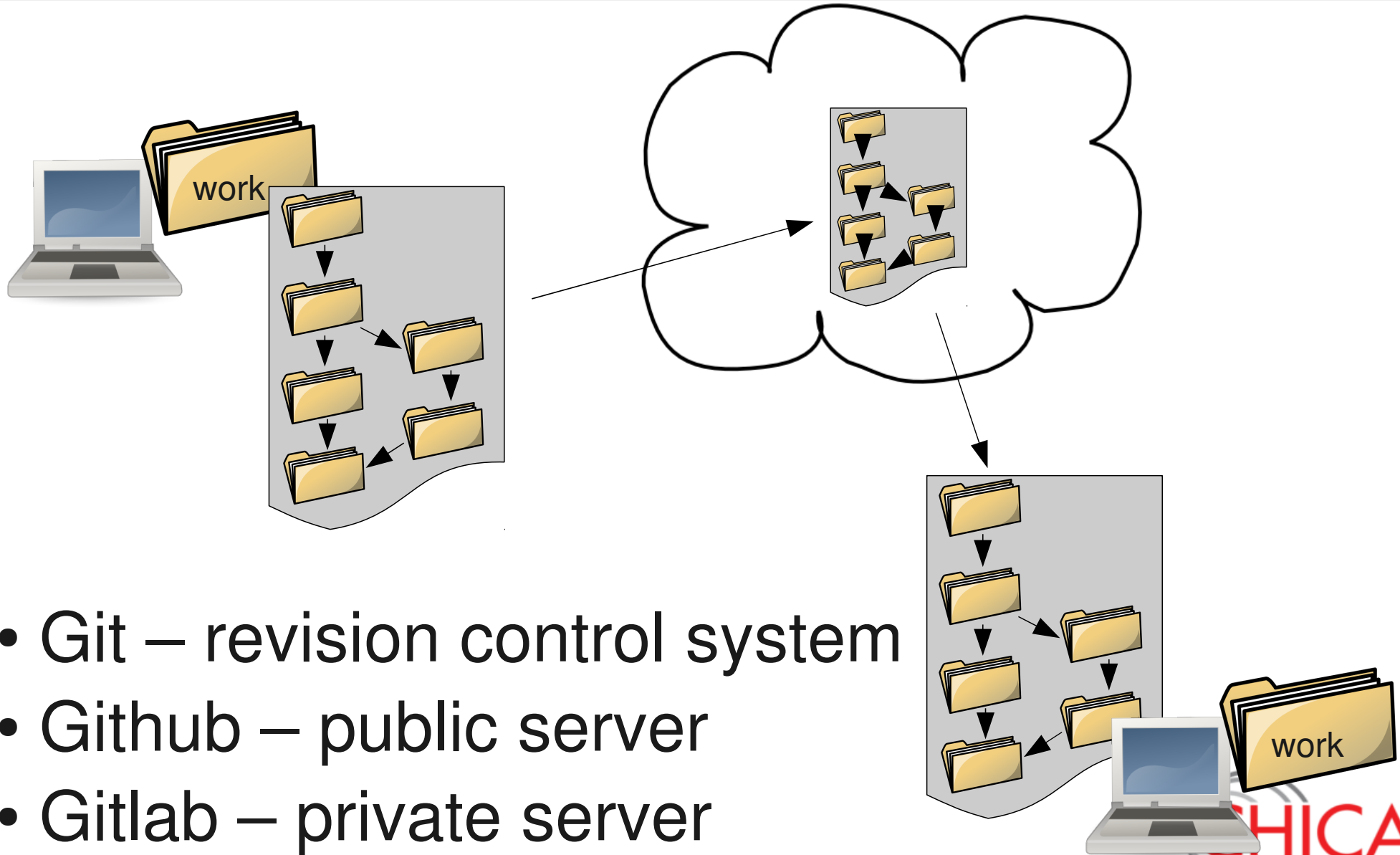
And so on...



branching/merging



collaborating



- Git – revision control system
- Github – public server
- Gitlab – private server

Time to move on

