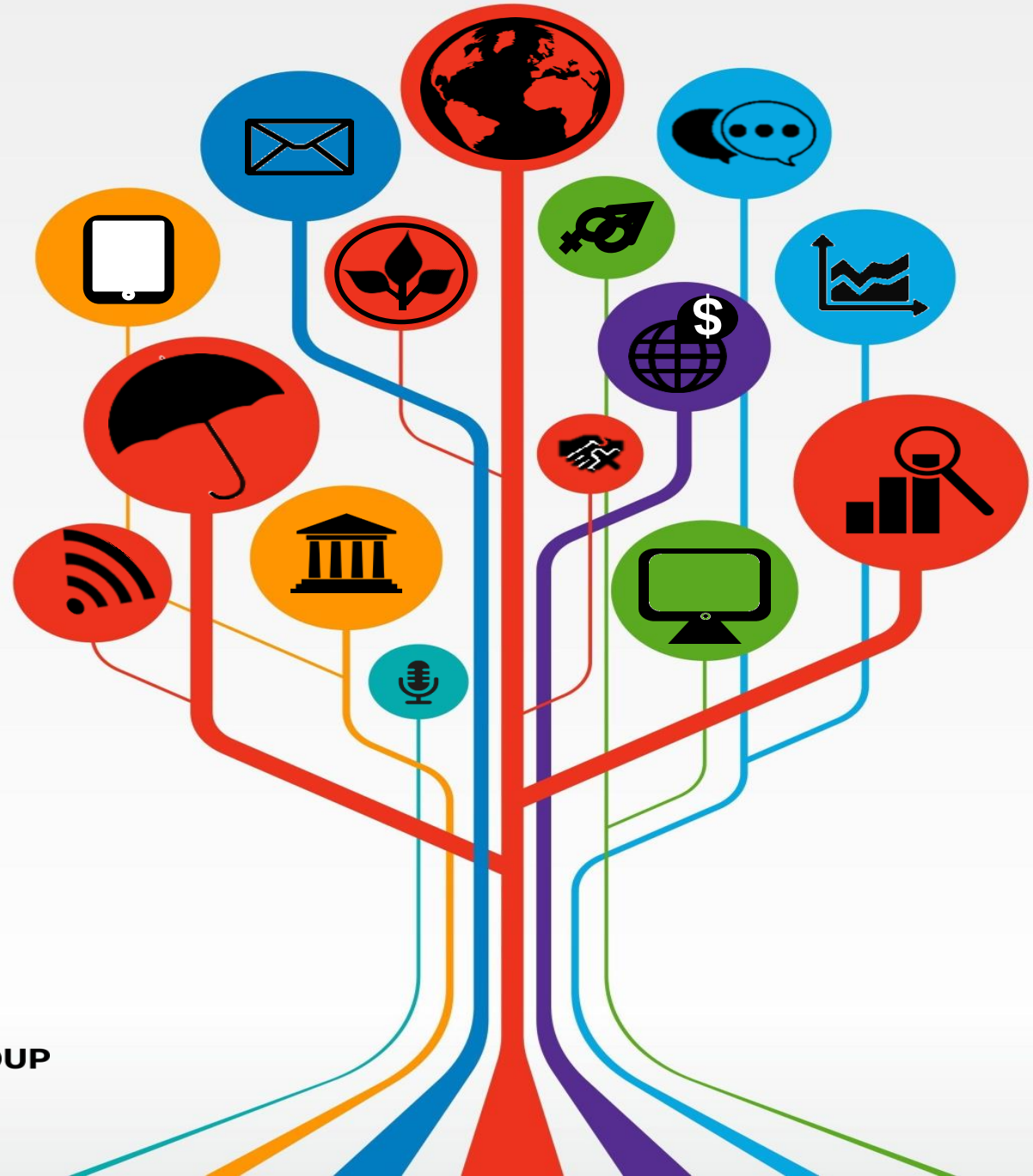
GitHub Training
Transfer to GitHub

Kristoffer Bjarkefur

November 11, 2019



i2i DIME
TRANSFORM DEVELOPMENT

WORLD BANK GROUP

GitHub

- We assume basic knowledge in GitHub, so we will not cover this today
- We will go through a step-by-step guide for how to transfer a project from, for example, DropBox to GitHub.
  - Not everything on DropBox will be transferred – mostly only code
- There are plenty of people in this room to help with this

GitHub usage

- Who is already using GitHub as the only way to collaborate on code in their project?
- Who is already using GitHub as one way to collaborate on code in their project?
- Who has used GitHub for anything in their project more than just setting up the repo?
  - For example written some documentation, or transferred some files there.
- Who has just set up the repo and done nothing with it?

Section 1: Transfer to Git/GitHub

First step: Set up your repo and clone

Who have access to the project repo?

We give access to repos through teams.

One of you will be the team maintainer and that person is responsible to add PIs and other team members, and remove people as they leave

Clone your repo

1. Go to your repo on github.com/worldbank

2. Clone it to your computer
   - Where is a good place and where is a bad place to create a clone?

.gitignore

- Some files should not be shared on GitHub
  - Efficiency reasons – binary files (.dta, .doc, .pdf, .png) slows down Git
  - Data security reasons – GitHub is secure, but not that secure
- Files not shared on GitHub, you can keep sharing on DropBox

- How to make sure these files are not shared on GitHub?
  - Do not put them in the clone – How long until anyone screws up?
  - Use a **.gitignore** file – we have a template for this. It tells Git which files, or which types of files, to share and not to share on GitHub

.gitignore

- Examples of ways you can ignore files:
  - `*.dta` or `*.doc` or `*.xls` etc. ignores all files of those formats
  - `data/` ignores all files in that folder and all its subfolders
  - `password.*` or `data.*` ignores all files with those names regardless of format
- Data can come in so many different formats? How can I make sure that I remember all those formats?
  - Ignore everything and then un-ignore the formats you do want to share on GitHub like this `!*.do`, `!*.R`, `!*.tex`
  - To correctly *ignore everything* and then *un-ignore selectively* requires a very specific setup. Use the DIME Template!!

Set up DIME Analytics .gitignore template

1. Go to https://github.com/worldbank/DIMEwiki/tree/master/Topics/GitHub
2. Click **gitignore_template.txt** then click **Raw**
3. In another tab in your browser, open the main page in your clone and click **Create new file**
4. In the "*Name your file...*" field, type **.gitignore** but do not press anything else
5. A button will show up that says **Choose .gitignore:** then click that button and select any template, it does not matter which one. Do not commit yet.
6. Replace the content from the template GitHub gave file with the content in **gitignore_template.txt** in the first tab. Then commit this file!
7. Now pull this commit to your clone on your laptop!

Section 1: Transfer to Git/GitHub

Second step: Move your files

Before we start…

- Before moving any files, make sure that the .gitignore file you created in your browser is pulled, i.e. synced, to your laptop

- Move the full data-work folder or move files selectively?
  - The best way to make sure that you do not forget to move anything is that you move everything and let your **.gitignore** do most of the job in deciding what to share
  - If your data-work folder is too big – so that it takes too long time to move or there simply isn't room for it on the computer – then you will have to move files selectively. If you do this make sure you get all files and get the folder structure right!

Move the files

1. **Copy** the data-work folder from DropBox (or wherever you currently share them) to your clone. Wait until this is done.
2. Go to your GitHub Desktop (or other Git client) and see all the files that are ready to commit.
   - DO **NOT** COMMIT THEM ALL AT ONCE!!
   - All files are sorted by the folder they are in. And the folders are ordered alphabetically.
   - This makes it possible to commit files folder by folder, which we recommend
   - Therefore go carefully through each folder and commit the files folder by folder, ensuring that no identifying information is leaked to GitHub

Section 1: Transfer to Git/GitHub

Third step: Re-clone your clone

Delete files not to be committed

- Delete files not committed!
  - Files that are not committed are files that you will keep sharing in DropBox.
- The easiest way to make sure you delete all files in your clone not shared on GitHub, is to delete the whole clone and then re-clone
- Deleting and re-cloning makes sure that you have the exact same files in your clone, as your team members when they clone this repo
- So after testing, which is next step, you know that if you can run the content in your clone, so can your team members

Delete your clone

- Go to your clone and delete it.
  - Do not just delete the content, delete the clone
  - You delete the clone by deleting the top-folder where we created the .gitignore

- Once you have deleted it. Clone it again. The same way you cloned it before the transfer

Section 1: Transfer to Git/GitHub

DONE!

Section 2: Make code works in Git/GitHub

Fourth step: Update folder paths in code

Update your root folders

- When all files and folders for a project are shared only in DropBox, your master script only needs one top-folder path
  - `global dropbox “C:/Users/username/Dropbox/DIME Analytics”`

- Now when you share code in GitHub and other files in DropBox you need two top-folder paths
  - `global dropbox “C:/Users/username/Dropbox/DIME Analytics”`
  - `global clone “C:/Users/username/Documents/GitHub/dimeanalytics”`

Your code will point to different locations

- Your code will point to
  - code in the clone
  - data in DropBox

```
*Define globals
global dropbox “C:\Users/username/Dropbox/DIME Analytics”
global clone “C:\Users/username/Documents/GitHub/dimeanalytics”

*Open data set
use “${dropbox}/Baseline/data/hh_data_deidentified.dta”, clear

*Run analysis file
do “${clone}/Baseline/dofiles/hh_analysis.do”
```

One error you all will face

- Not all folders that your code require will be in your clone!
- Don’t worry, DIME Analytics has a solutions, but one step at the time

Add the clone folder path

1. In your master scripts, add a clone top-folder path:

Old ->

```
if ${user} == 1 {
    global projectfolder “C:/Users/username/Dropbox/DIME Analytics”
}
```

New ->

```
if ${user} == 1 {
    global dropbox “C:/Users/username/Dropbox/DIME Analytics”
    global clone “C:/Users/username/Documents/GitHub/dimeanalytics”
}
```

2. Update other globals:

```
global Baseline     "${projectfolder}/Baseline"
global Baseline_dt  "${Baseline}/DataSets"
global Baseline_do  "${Baseline}/Dofiles"
global Baseline_out "${Baseline}/Output"
```

<- Old

New ->

```
global Baseline_db  "${dropbox}/Baseline"
global Baseline     "${clone}/Baseline"
global Baseline_dt  "${Baseline_db}/DataSets"
global Baseline_do  "${Baseline}/Dofiles"
global Baseline_out "${Baseline}/Output"
```

Section 2: Make code work in Git/GitHub

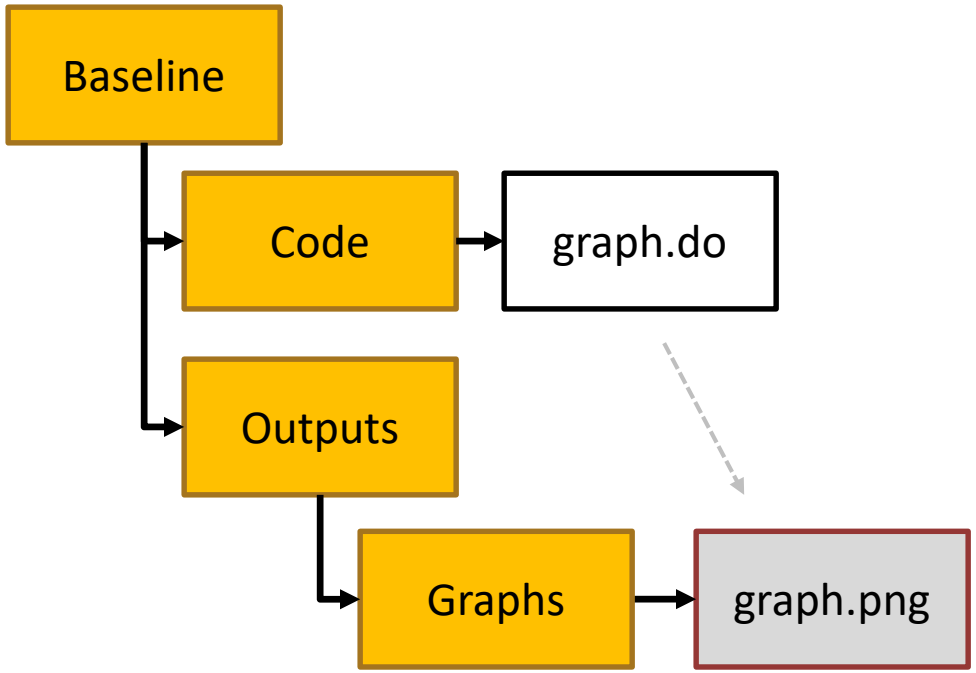
Fifth step: Automatic fix to empty folders issue

One error you all will have to fix

- In Git nothing but files are shared
- Folders are only implicitly shared in Git as files are shared with the folder location in the file name
  - Example: src/ado_files/iebaltab.ado is the full filename to Git
  - So when you share a file, you also share the folder that it is saved in, and all its parent folders
- Folders in with no files that are shared, or with no sub-folders with files that are shared, are therefore not shared in Git

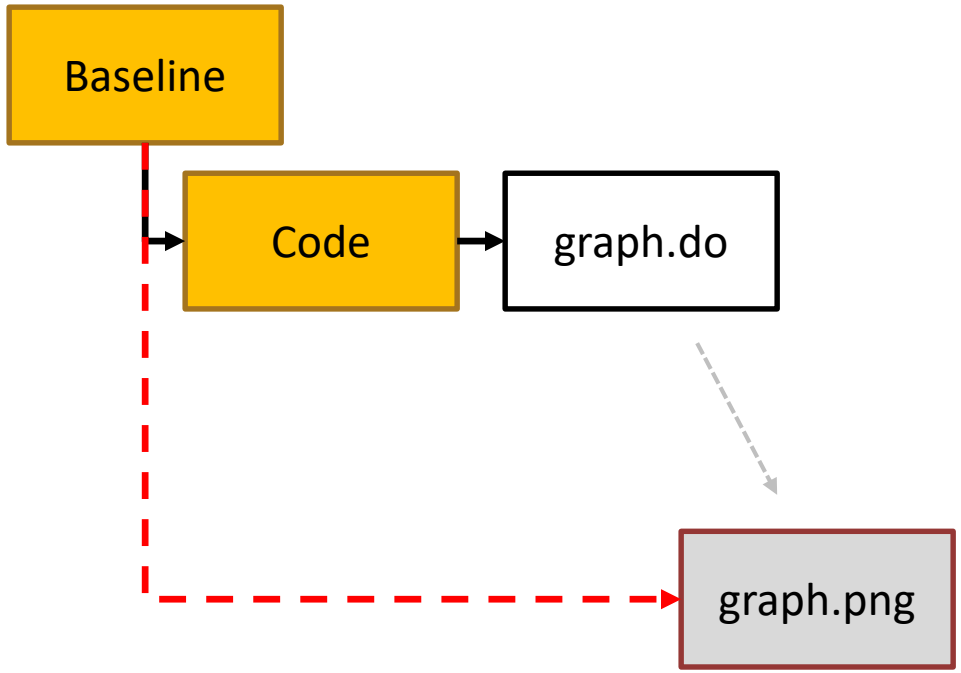
Empty folders – the problem

RA-clone



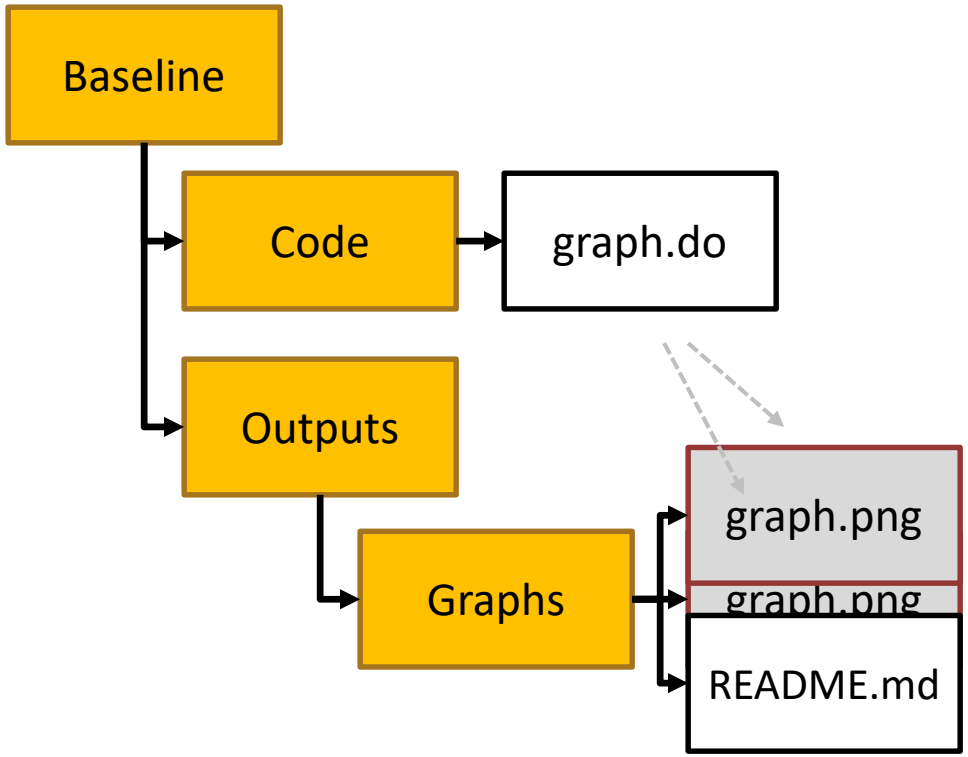
.png files are ignored in this repository

PI-clone



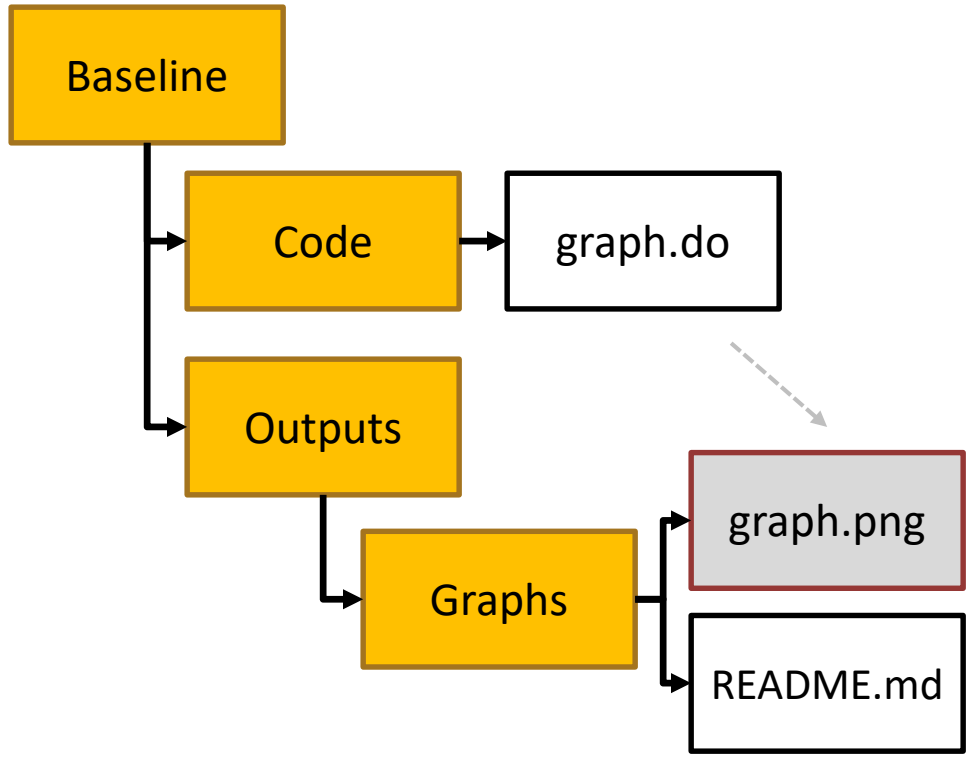
Empty folders – the solution

RA-clone



.png files are ignored in this repository

PI-clone



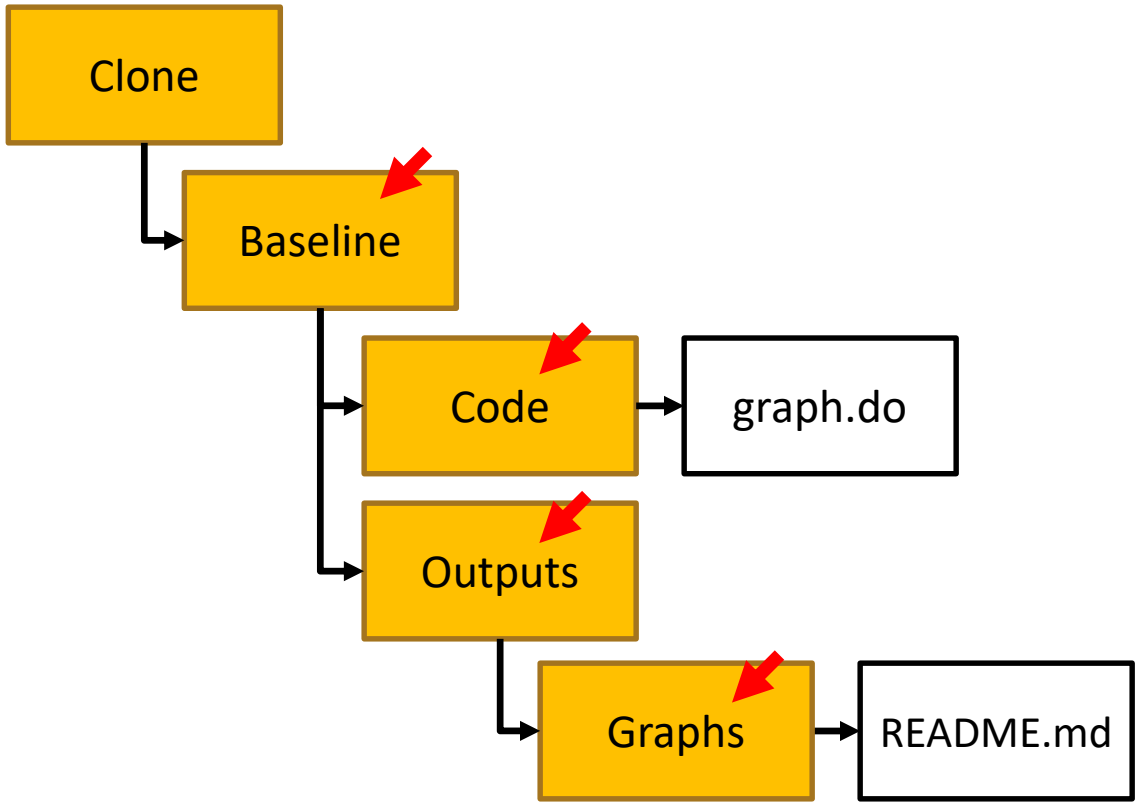
Manual fix

- https://github.com/worldbank/DIMEwiki/tree/master/Topics/GitHub

- But lets focus on the more convenient way

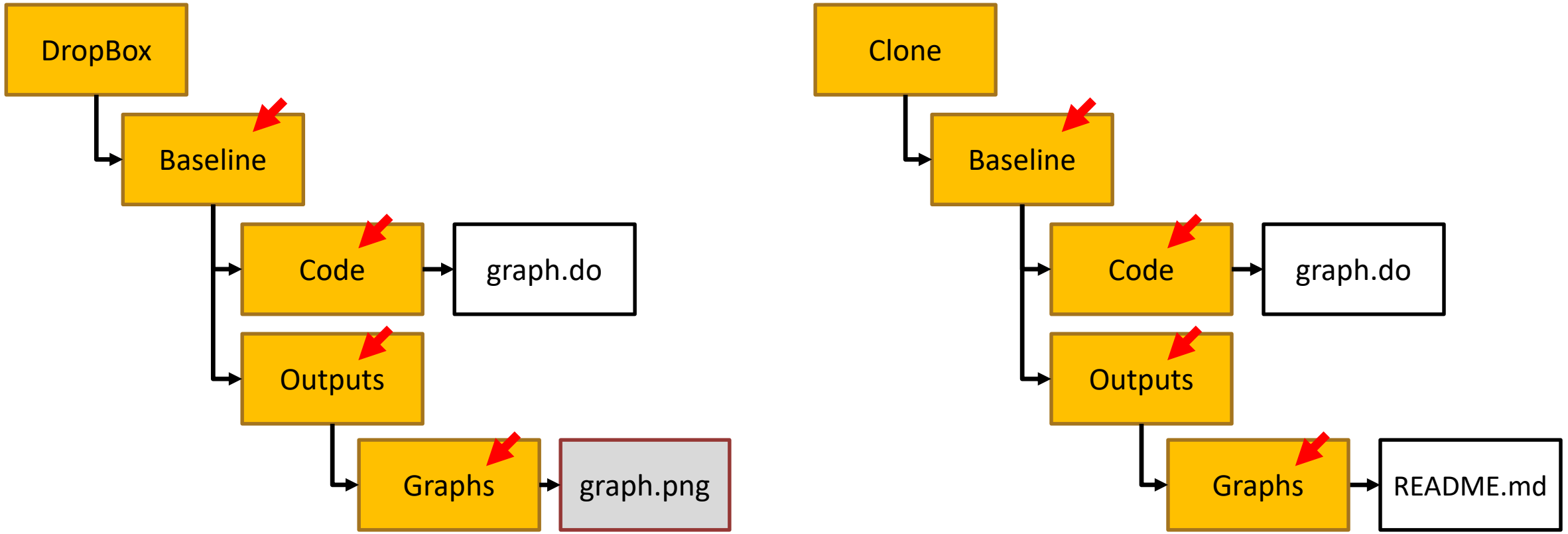
iegitaddmd – use case 1

```
iegitaddmd, folder("${clone}/Baseline")
```



iegitaddmd – use case 2

```
iegitaddmd, folder("${clone}/Baseline") comparefolder("${dropbox}/Baseline")
```



iegitaddmd – best practice

1. Use `dryrun` option to test that folder paths are correctly specified. Makes the command list where placeholder files would have been created if the `dryrun` option was not used

```
iegitaddmd, folder(“${clone}/Baseline”) comparefolder(“${dropbox}/Baseline”) dryrun
```

```
iegitaddmd, folder(“${clone}/Baseline”) dryrun
```

2. Then remove `dryrun` option to start creating placeholder files

3. Once you understand how the command works and you feel comfortable that you have specified the correct path, consider using the `automatic` option

4. Now test if your master script runs again!

Section 2: Make code work in Git/GitHub

Sixth step: Delete code in DropBox

(can be done later)

Do not store code in two locations

- Storing something in two places will cause confusion. So don’t do it
- But if you want to wait a week to do so, while you are still testing that everything makes sense, then it is fine to do so
- Please let is know what you think has been most difficult getting started with Git up until today, during today’s session or at any point in the future