Reproducible and Interpretable Data Management:
Looking Out For Future You (And Other People Too)

Sarah Odell
University of California Davis
January 13, 2018

https://bit.ly/2CfbXLu

Reproducibility Is A Spectrum

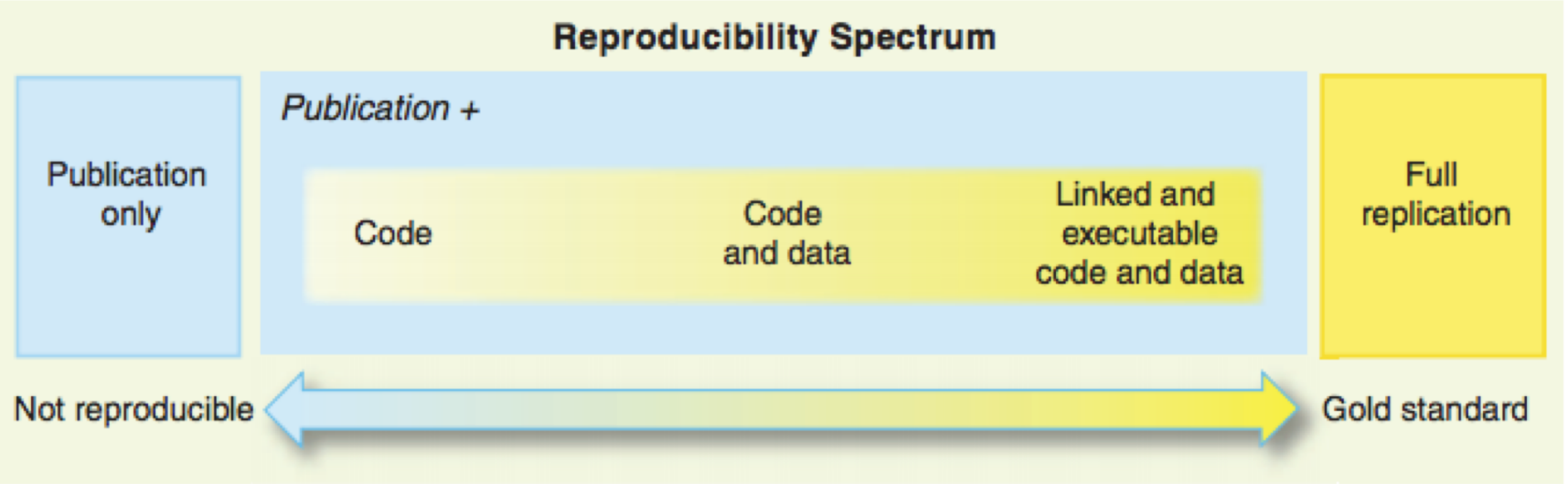


Fig. 1. The spectrum of reproducibility.

Peng, 2011

Reproducibility Is A Spectrum

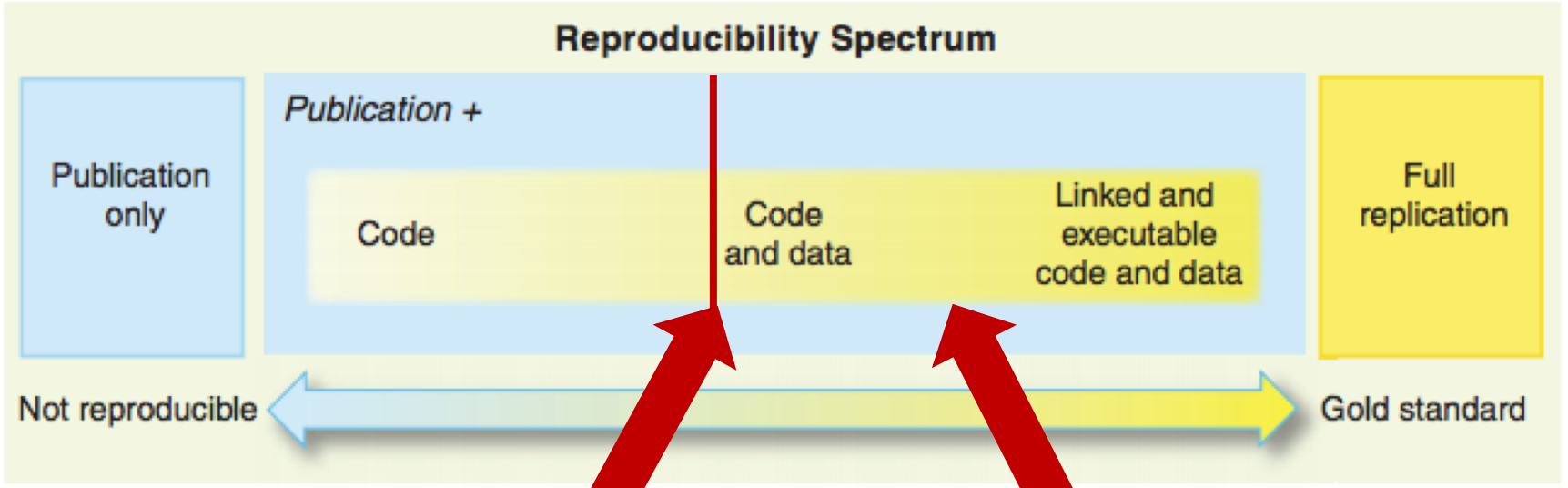


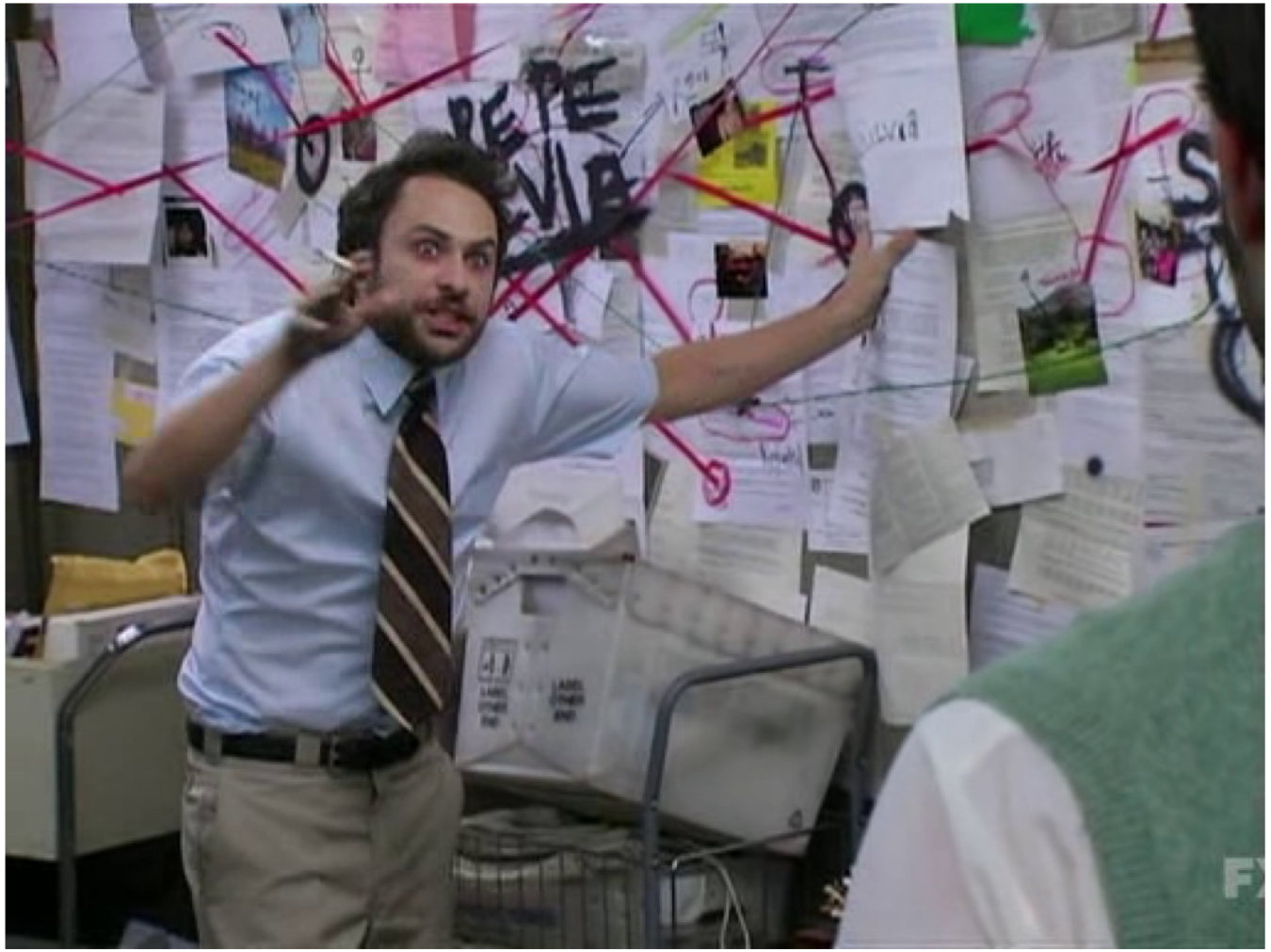
Fig. 1. The spectrum of reprodu

Baseline

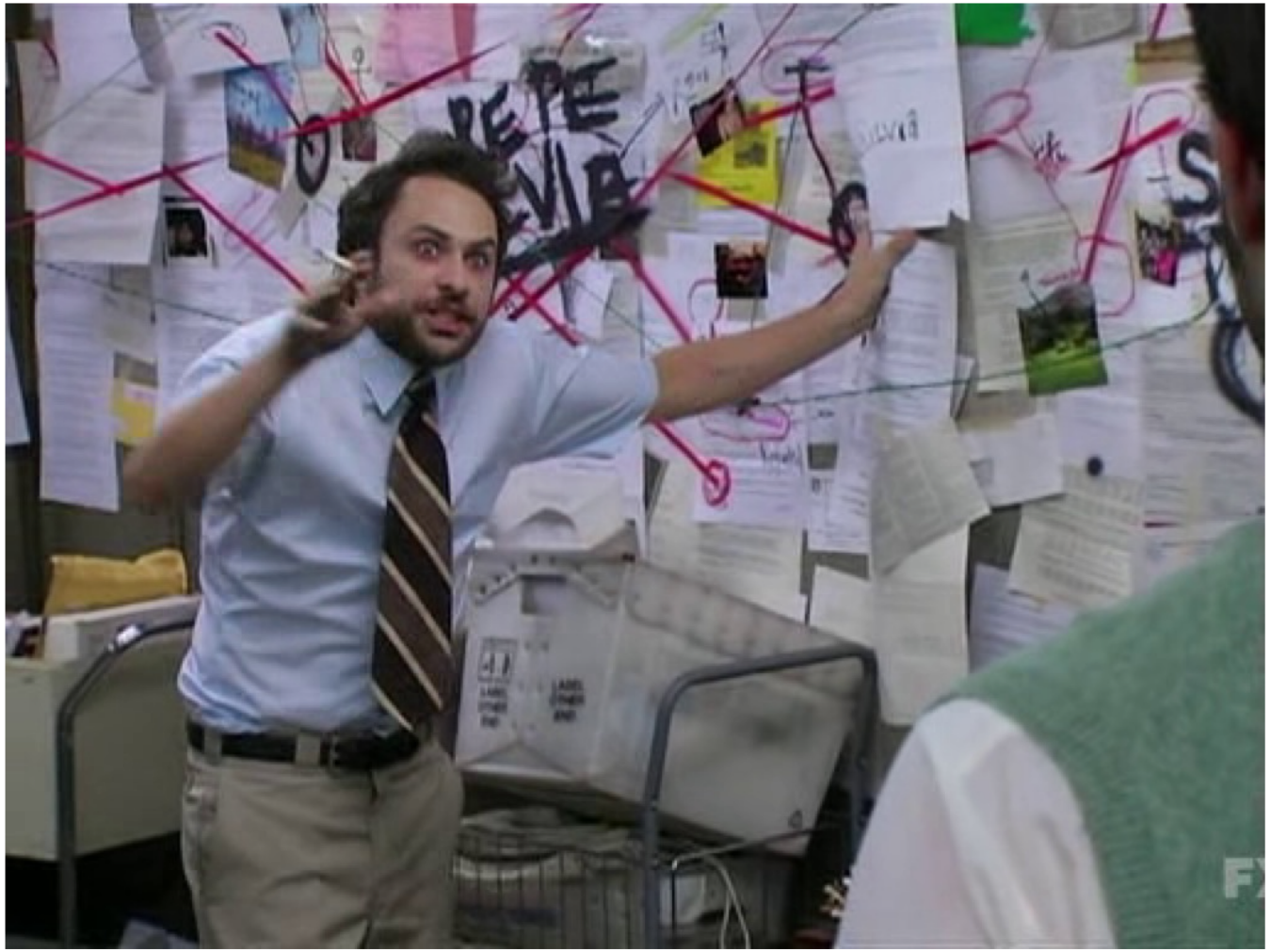
For us mere mortals

Peng, 2011

Trying to explain how you did your analysis to a collaborator in six months



Trying to explain how you did your analysis to **yourself** in six months



Start

Main Topics

1. Naming Things
2. Version Control
3. Making Code for Humans
4. Building Workflows

Finish

Naming Things

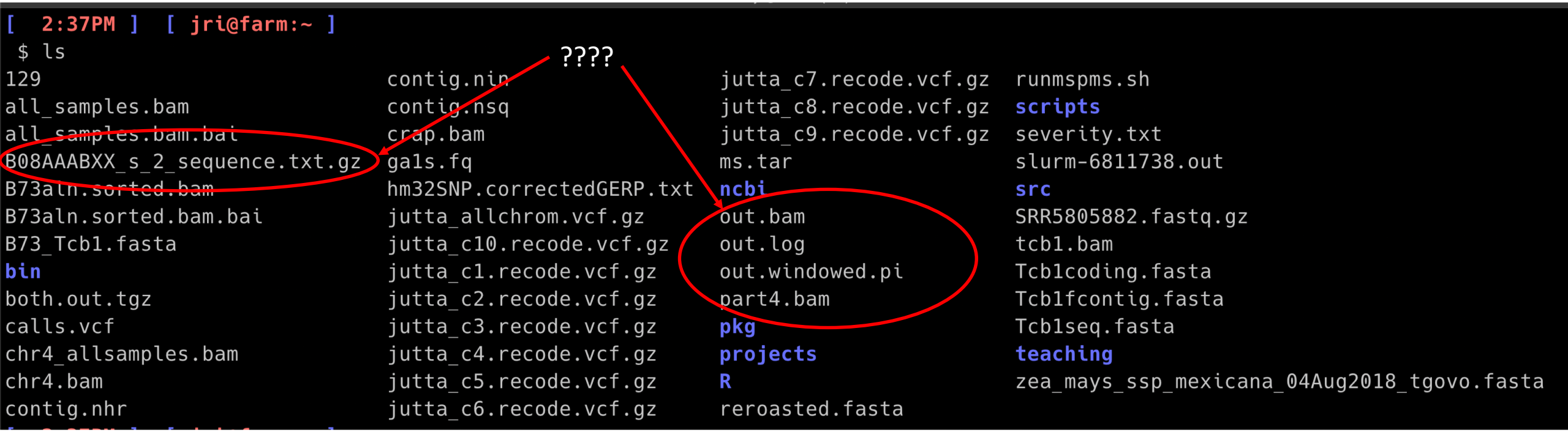
- Descriptive data files and script names

- The final is not the final (I use dates)

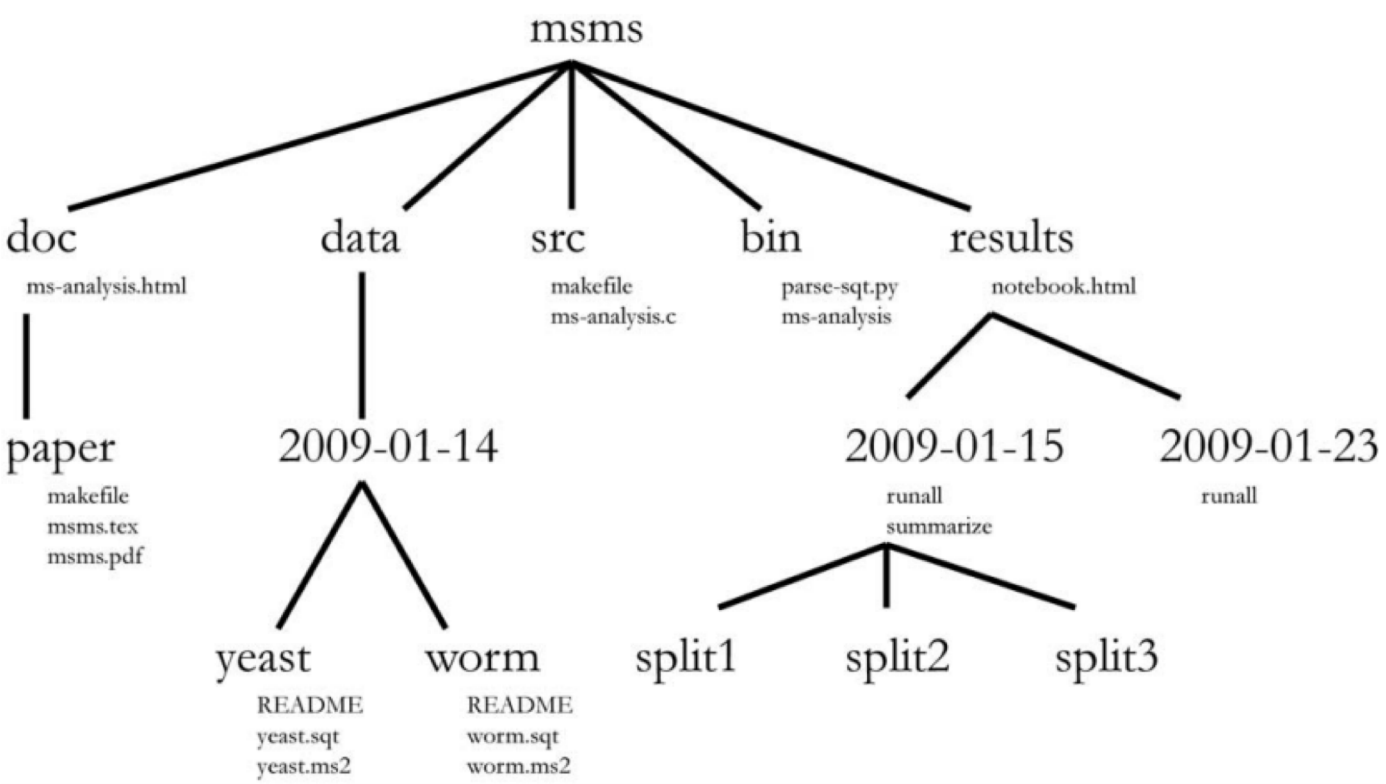
```
[  2:37PM ]  [ jri@farm:~ ]
 $ ls
129                           contig.nin                jutta_c7.recode.vcf.gz  runmspms.sh
all_samples.bam               contig.nsq                jutta_c8.recode.vcf.gz  scripts
all_samples.bam.bai           crap.bam                  jutta_c9.recode.vcf.gz  severity.txt
B08AAABXX_s_2_sequence.txt.gz ga1s.fq                   ms.tar                  slurm-6811738.out
B73aln.sorted.bam             hm32SNP.correctedGERP.txt ncbi                    src
B73aln.sorted.bam.bai         jutta_allchrom.vcf.gz     out.bam                 SRR5805882.fastq.gz
B73_Tcb1.fasta                jutta_c10.recode.vcf.gz   out.log                 tcb1.bam
bin                           jutta_c1.recode.vcf.gz    out.windowed.pi         Tcb1coding.fasta
both.out.tgz                  jutta_c2.recode.vcf.gz    part4.bam               Tcb1fcontig.fasta
calls.vcf                     jutta_c3.recode.vcf.gz    pkg                     Tcb1seq.fasta
chr4_allsamples.bam           jutta_c4.recode.vcf.gz    projects                teaching
chr4.bam                      jutta_c5.recode.vcf.gz    R                       zea_mays_ssp_mexicana_04Aug2018_tgovo.fasta
contig.nhr                    jutta_c6.recode.vcf.gz    reroasted.fasta
```

Naming Things

- Descriptive data files and script names
- The final is not the final (I use dates)



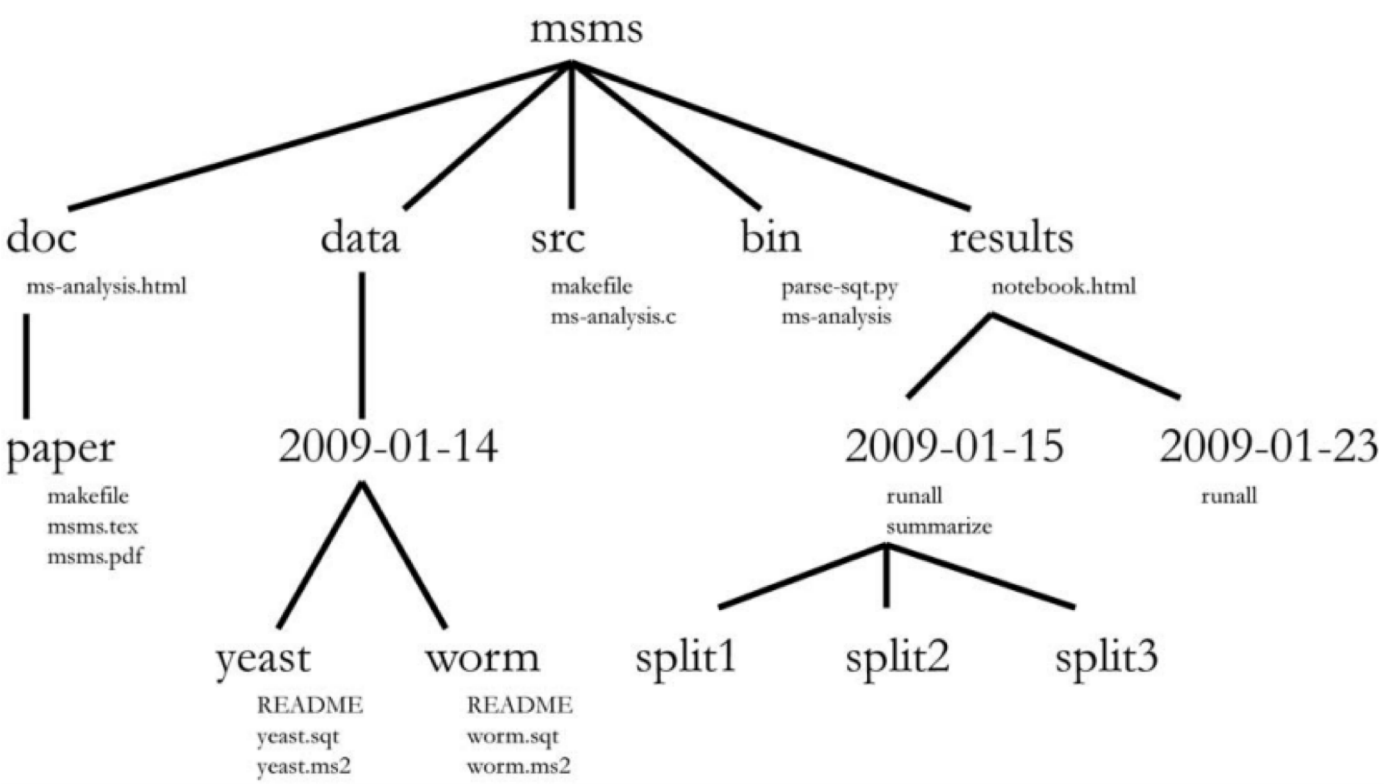
Directory Organization



- Keep related files in the same directory
- Make a README for that directory (and each internal directory)

Noble, 2009

Directory Organization



- Keep related files in the same directory
- Make a README for that directory (and each internal directory)
- Can use a data science project cookiecutter to start
- Keep an untouched copy of your raw data

Noble, 2009

https://drivendata.github.io/cookiecutter-data-science/

Start

Main Topics

1. Naming Things
2. Version Control
3. Making Code for Humans
4. Building Workflows

Finish

Version Control

Image Source: http://ipengineer.net/2015/04/git-for-network-engineers/

Version Control

Branch: master ▾ impute / pscripts /

Create new file | Upload files | Find file | History

sarahodell Write definitions — Latest commit b2605db 16 minutes ago

..		
PCA_600K.Rmd	Write definitions	16 minutes ago
README.md	PCA markdown	4 months ago
approx_cM.py	fixed some bugs	6 months ago
build_ril.py	Organized	6 months ago
build_simvcf.py	fixed some bugs	6 months ago

Image Source: http://ipengineer.net/2015/04/git-for-network-engineers/

Version Control

Showing 4 changed files with 56 additions and 41 deletions.

20 pscripts/approx_cM.py

```
@@ -36,16 +36,18 @@ def approx_cM(chrom,start,end,ref='v4'):
36 36       cmap=ogutmap[ogutmap['chr']==chrom]
37 37       cmap.reset_index(inplace=True)
38 38       #Closest ogut map markers above and below the start and end postions
39    -     below=cmap.iloc[cmap[cmap['pos']<=start]['pos'].idxmax(),]
40    -     above=cmap.iloc[cmap[cmap['pos']>=end]['pos'].idxmin(),]
41    -     xpoints=[below['pos'],above['pos']]
42    -     ypoints=[below['cM'],above['cM']]
43    -     #Linear regression of genetic position on physical position using two closest points
44    -     slope,intercept,r_value,p_value,stderr = stats.linregress(xpoints,ypoints)
45    -     start_cM=slope*start + intercept
46    -     end_cM=slope*end + intercept
   39 +     dist=[]
   40 +     for p in [start,end]:
   41 +         below=cmap.iloc[cmap[cmap['pos']<=p]['pos'].idxmax(),]
   42 +         above=cmap.iloc[cmap[cmap['pos']>=p]['pos'].idxmin(),]
   43 +         xpoints=[below['pos'],above['pos']]
   44 +         ypoints=[below['cM'],above['cM']]
```

Branch: master ▾ impute / pscripts /

sarahodell Write definitions

..		
PCA_600K.Rmd		
README.md	PCA markdown	4 months ago
approx_cM.py	fixed some bugs	6 months ago
build_ril.py	Organized	6 months ago
build_simvcf.py	fixed some bugs	6 months ago

Image Source: http://ipengineer.net/2015/04/git-for-network-engineers/

Start

Main Topics

1. Naming Things
2. Version Control
3. Making Code for Humans
4. Building Workflows

Finish

Making Code for Humans

```
pr = readRDS('../../qtl2/MAGICSim_062018/geno_probs.rds')
apr = readRDS('../../qtl2/MAGICSim_062018/allele_probs.rds')
pmap10 = read.csv('../../qtl2/MAGICSim_062018/chr10/MAGICSim_pmap_c10.csv')
founders=c("A632_usa","B73_inra","CO255_inra","FV252_inra","OH43_inra","A654_inra",
           "FV2_inra","C103_inra","EP1_inra","D105_inra","W117_inra","B96","DK63",
           "F492","ND245","VA85")
pr_c <- clean_genoprob(pr,value_threshold = 0.0001)
all_pr=c()
for (m in 1:10){
  md <- pr_c[[1]][m,,]
  tm <- t(md)
  mdf <- as.data.frame(tm,row.names = rownames(tm))
  mdf<-rownames_to_column(mdf,"marker")
  names(mdf)=c("marker",founders)
  mdf <- merge(mdf,pmap10,by.x='marker',by.y='marker')
  mdf <- mdf[,c(2:17,19)]
  mlong<-melt(mdf,id="pos")
  mlong$sample=rep(sprintf("M%s",m),dim(mlong)[1])
  all_pr=rbind(all_pr,mlong)
}
```

Making Code for Humans

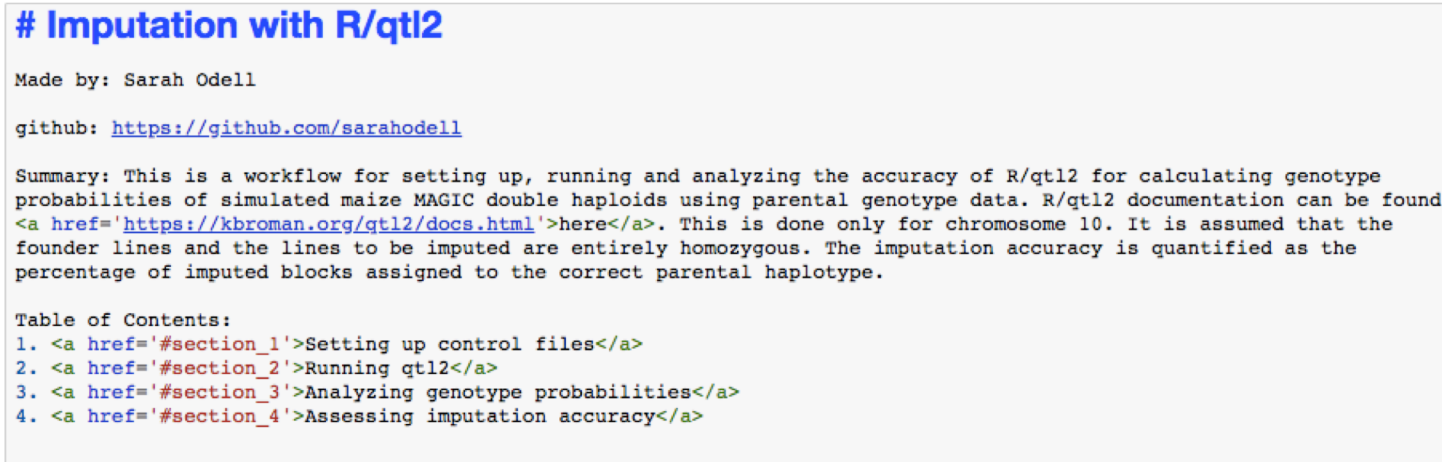
```
pr = readRDS('../../qtl2/MAGICSim_062018/geno_probs.rds')
apr = readRDS('../../qtl2/MAGICSim_062018/allele_probs.rds')
pmap10 = read.csv('../../qtl2/MAGICSim_062018/chr10/MAGICSim_pmap_c10.csv')
founders=c("A632_usa","B73_inra","CO255_inra","FV252_inra","OH43_inra","A654_inra",
           "FV2_inra","C103_inra","EP1_inra","D105_inra","W117_inra","B96","DK63",
           "F492","ND245","VA85")
pr_c <- clean_genoprob(pr,value_threshold = 0.0001)
all_pr=c()
for (m in 1:10){
  md <- pr_c[[1]][m,,]
  tm <- t(md)
  mdf <- as.data.frame(tm,row.names = rownames(tm))
  mdf<-rownames_to_column(mdf,"marker")
  names(mdf)=c("marker",founders)
  mdf <- merge(mdf,pmap10,by.x='marker',by.y='marker')
  mdf <- mdf[,c(2:17,19)]
  mlong<-melt(mdf,id="pos")
  mlong$sample=rep(sprintf("M%s",m),dim(mlong)[1])
  all_pr=rbind(all_pr,mlong)
}
```

```
wkdir='~/sodell/Documents/impute'
setwd(wdkir) # Set the working directory to wherever the local project directory is

pr=readRDS(sprintf('%s/data_files/geno_probs.rds',wkdir)) # Genotype probabilities

pmap10 = read.csv(sprintf('%s/data_files/Maize_pmap_c10.csv',wkdir)) # Chr 10 physical map

founders=c("A632_usa","B73_inra","CO255_inra","FV252_inra","OH43_inra","A654_inra",
           "FV2_inra","C103_inra","EP1_inra","D105_inra","W117_inra","B96","DK63",
           "F492","ND245","VA85") #Names of the population founder lines
```

We now want to clean the genotype probabilities so that very small values are set to 0, and large values set to 1. Then, restructure the pr object so that it can be easily plotted.

```
pr_clean <- clean_genoprob(pr,value_threshold = 0.0001) # Clean the genotype probs
all_pr=c()
for (m in 1:10){ # for each sample
  sample <- pr_clean[[1]][m,,]
  tsample <- t(sample)
  #grab that sample from pr_clean, and transpose the matrix
  tsampledf <- as.data.frame(tsample,row.names = rownames(tsample))
  tsampledf<-rownames_to_column(tsampledf,"marker")
  names(tsampledf)=c("marker",founders)
  tsampledf <- merge(tsampledf,pmap10,by.x='marker',by.y='marker')
  #merge the probabilities and the physical map data by marker name
  tsampledf <- tsampledf[,c(2:17,19)]
  mlong<-melt(tsampledf,id="pos")
  #melt the table so that it is grouped by physical position
  mlong$sample=rep(sprintf("M%s",m),dim(mlong)[1])
  # add a sample name column
  all_pr=rbind(all_pr,mlong)
}
```

Making Code for Humans – Jupyter Notebooks



Imputation with R/qtl2

Made by: Sarah Odell

github: https://github.com/sarahodell

Summary: This is a workflow for setting up, running and analyzing the accuracy of R/qtl2 for calculating genotype probabilities of simulated maize MAGIC double haploids using parental genotype data. R/qtl2 documentation can be found here. This is done only for chromosome 10. It is assumed that the founder lines and the lines to be imputed are entirely homozygous. The imputation accuracy is quantified as the percentage of imputed blocks assigned to the correct parental haplotype.

Table of Contents:

1. Setting up control files
2. Running qtl2
3. Analyzing genotype probabilities
4. Assessing imputation accuracy

- Can show Markdown, Python, R and command line
- Can be converted into html, pdf, etc.

Making Code for Humans – Jupyter Notebooks

```
# Imputation with R/qtl2

Made by: Sarah Odell

github: https://github.com/sarahodell

Summary: This is a workflow for setting up, running and analyzing the accuracy of R/qtl2 for calculating genotype
probabilities of simulated maize MAGIC double haploids using parental genotype data. R/qtl2 documentation can be found
<a href='https://kbroman.org/qtl2/docs.html'>here</a>. This is done only for chromosome 10. It is assumed that the
founder lines and the lines to be imputed are entirely homozygous. The imputation accuracy is quantified as the
percentage of imputed blocks assigned to the correct parental haplotype.

Table of Contents:
1. <a href='#section_1'>Setting up control files</a>
2. <a href='#section_2'>Running qtl2</a>
3. <a href='#section_3'>Analyzing genotype probabilities</a>
4. <a href='#section_4'>Assessing imputation accuracy</a>
```

Imputation with R/qtl2

Made by: Sarah Odell

github: https://github.com/sarahodell

Summary: This is a workflow for setting up, running and analyzing the accuracy of R/qtl2 for calculating genotype probabilities of simulated maize MAGIC double haploids using parental genotype data. R/qtl2 documentation can be found here. This is done only for chromosome 10. It is assumed that the founder lines and the lines to be imputed are entirely homozygous. The imputation accuracy is quantified as the percentage of imputed blocks assigned to the correct parental haplotype.

Table of Contents:

1. Setting up control files
2. Running qtl2
3. Analyzing genotype probabilities
4. Assessing imputation accuracy

- Can show Markdown, Python, R and command line
- Can be converted into html, pdf, etc.

“Soft Code” Paths and Variables

```
pmap10=read.csv('/Users/sodell/projects/impute/Maize_pmap_c10.csv')
```

```
Warning message in file(file, "rt"):
“cannot open file '/Users/sodell/projects/impute/Maize_pmap_c10.csv': No such file or directo
ry”
Error in file(file, "rt"): cannot open the connection
Traceback:

1. read.csv("/Users/sodell/projects/impute/Maize_pmap_c10.csv")
2. read.table(file = file, header = header, sep = sep, quote = quote,
 .     dec = dec, fill = fill, comment.char = comment.char, ...)
3. file(file, "rt")
```

“Soft Code” Paths and Variables

Set the working directory as a variable. It can point to the head directory of your project, so that all other files and scripts can be called within the project directory.

Read in all the files that you need. Can display a portion of the data.

```
wkdir='~/sodell/Documents/impute'
setwd(wdkir) # Set the working directory to wherever the local project directory is

pr=readRDS(sprintf('%s/data_files/geno_probs.rds',wkdir)) # Genotype probabilities
pmap10 = read.csv(sprintf('%s/data_files/Maize_pmap_c10.csv',wkdir)) # Chr 10 physical map

head(pmap)
```

marker	chr	pos
AX-91807321	10	0.275461
AX-91157290	10	0.276497
AX-91807318	10	0.276688
AX-91157288	10	0.279348
AX-91157305	10	0.301249
AX-90617680	10	0.301380

Date and Software Versions

Created On: 07/18/18

Last Updated: 09/21/18

Date and Software Versions

Created On: 07/18/18

Last Updated: 09/21/18

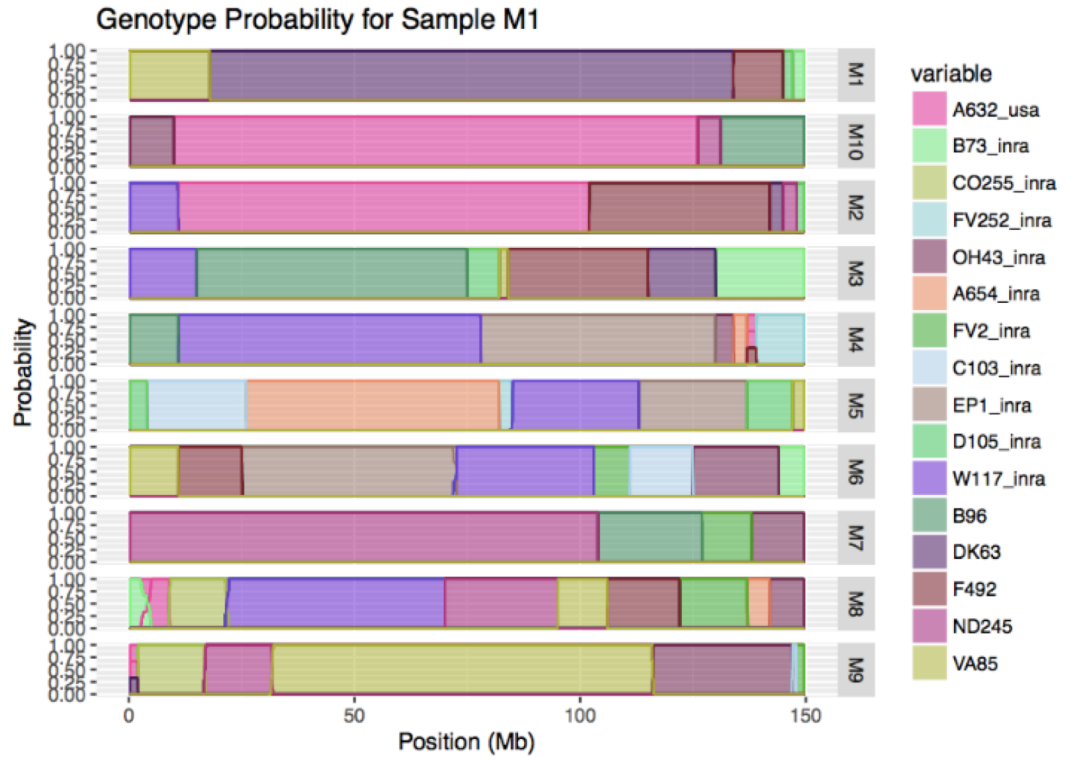
```
print("Created On: 07/18/18")
print(format(Sys.time(), "Last updated: %m/%d/%Y"))
print("Created Using R Version 3.4.2")
print(sprintf("Current Version: %s",R.Version()$version.string))
```

```
[1] "Created On: 07/18/18"
[1] "Last updated: 01/12/2019"
[1] "Created Using R Version 3.4.2"
[1] "Current Version: R version 3.4.2 (2017-09-28)"
```

Recreating Figures

- Provide code to recreate figures

```
In [ ]: ggplot(data=all_pr,aes(x=pos,y=value,color=variable)) + facet_grid(sample ~ .) +
scale_color_manual(values=hex_colors) + scale_fill_manual(values=hex_colors)+
geom_area(aes(fill=variable),alpha=5/10) + geom_line() +
ggtitle("Genotype Probability for Sample M1") + xlab("Position (Mb)") + ylab("Probability") +
guides(color=FALSE)
```



Recreating Figures

- Provide code to recreate figures
- Embed an image of the original for comparison

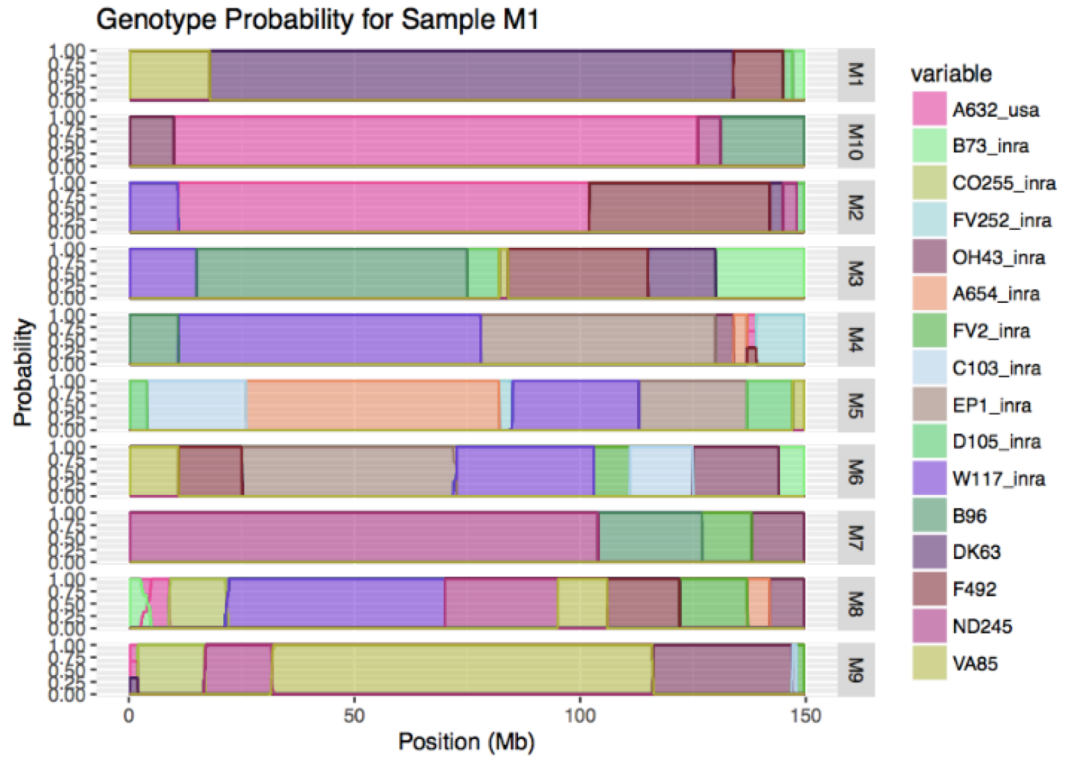
```
And here is a plot we created originally:

Original
![image](../images/qtl2_image.png)
```

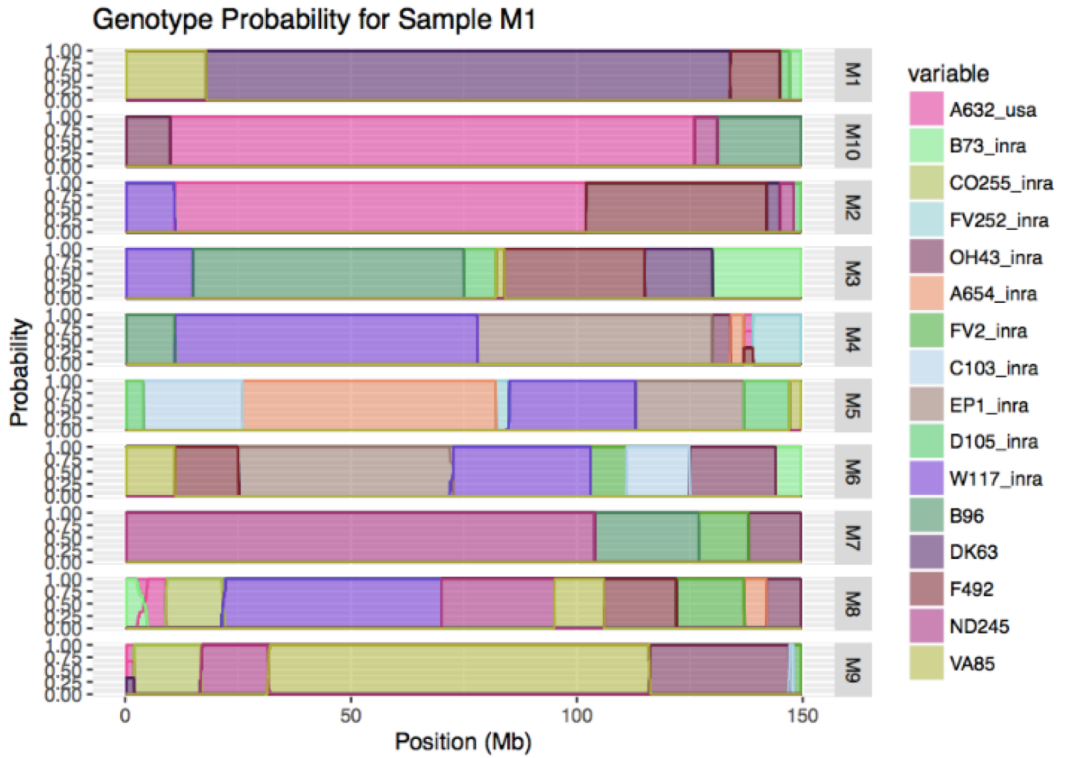
And here is a plot we created originally:

Original

In [ ]:
```
ggplot(data=all_pr,aes(x=pos,y=value,color=variable)) + facet_grid(sample ~ .) +
scale_color_manual(values=hex_colors) + scale_fill_manual(values=hex_colors)+
geom_area(aes(fill=variable),alpha=5/10) + geom_line() +
ggtitle("Genotype Probability for Sample M1") + xlab("Position (Mb)") + ylab("Probability") +
guides(color=FALSE)
```

In [ ]:





Start

Main Topics

1. Naming Things

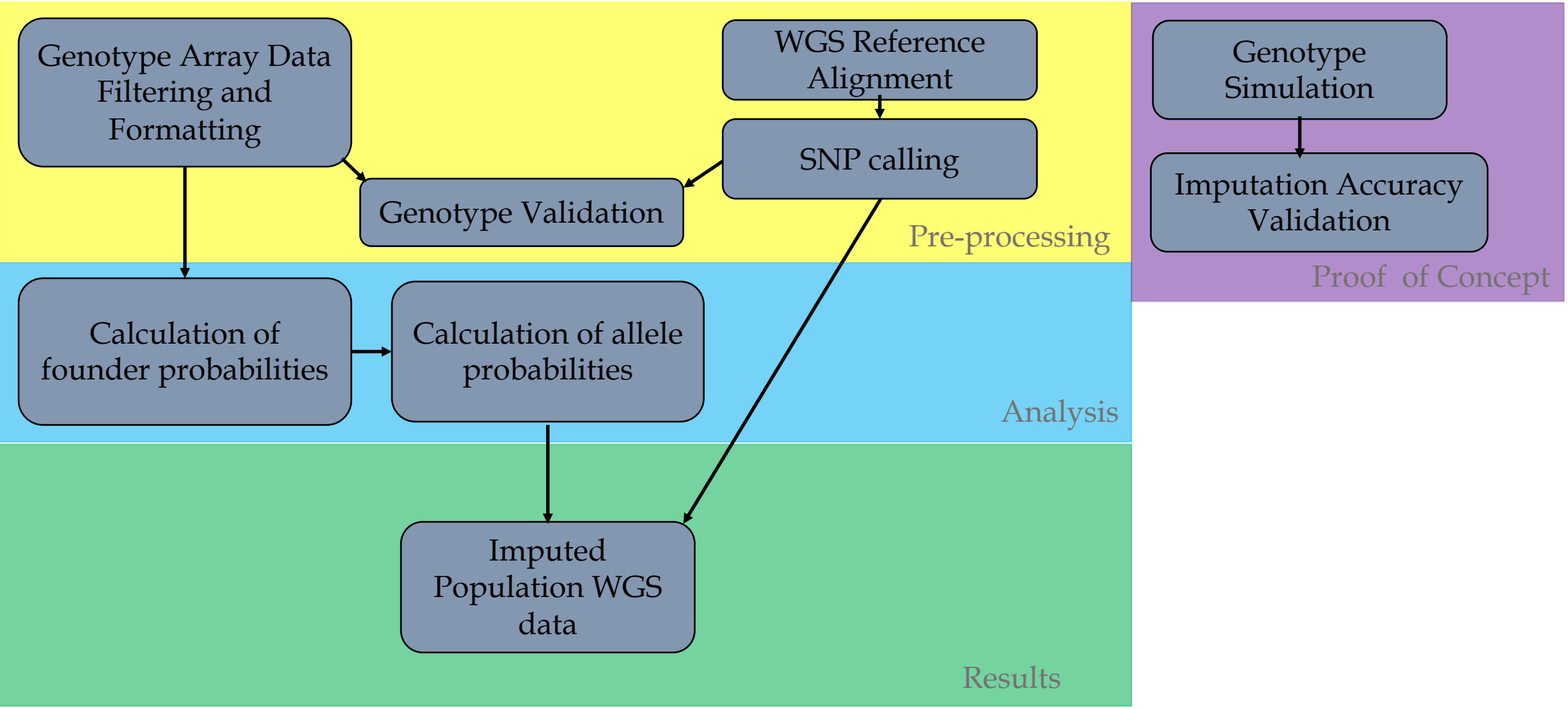
2. Version Control

3. Making Code for Humans

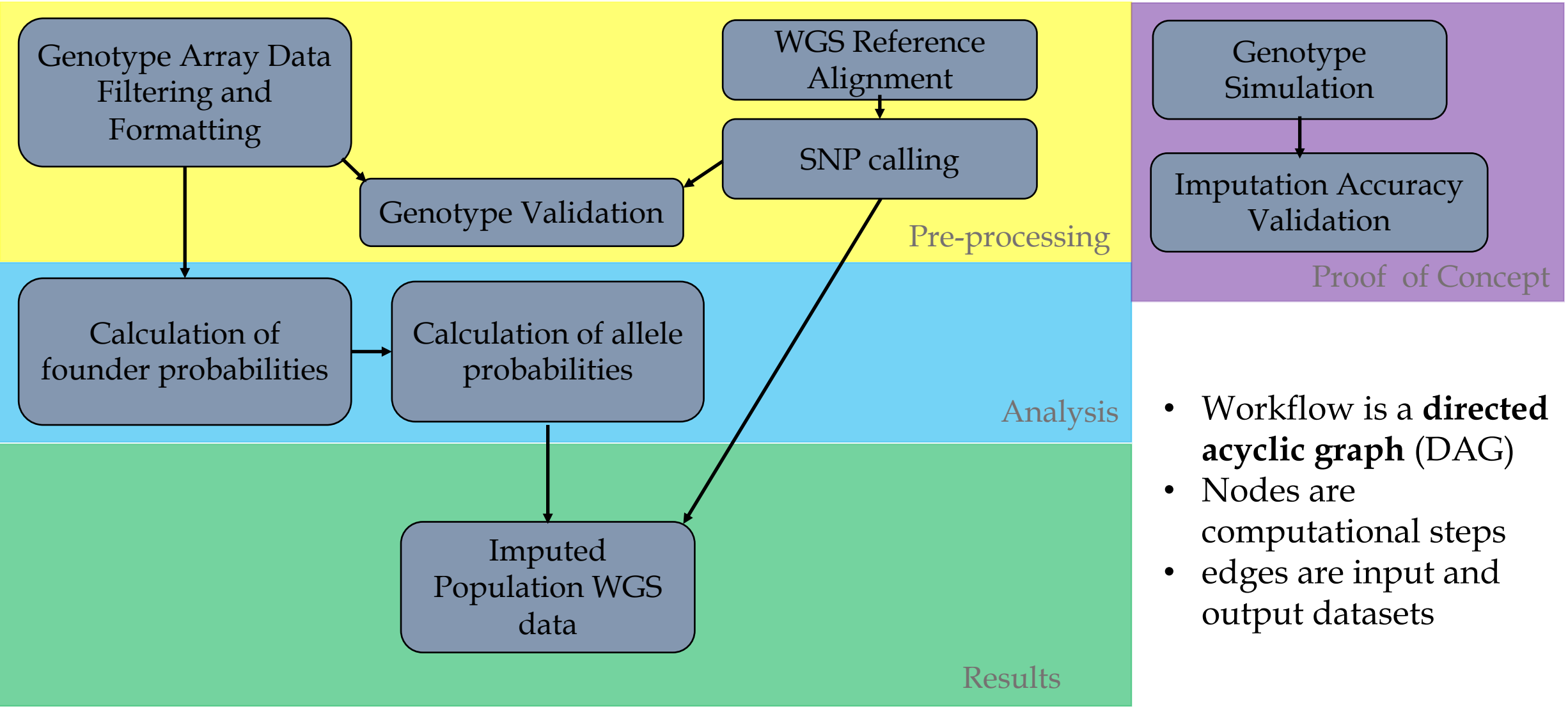
4. Building Workflows

Finish

Building Workflows



Building Workflows



- Workflow is a **directed acyclic graph** (DAG)
- Nodes are computational steps
- edges are input and output datasets

Building Workflows

The problems:

- Downloading and installing all the software, packages and dependencies that you need to run a workflow can be a nightmare

- Version updates can break, deprecate, or change how tools work

```
> install.packages('devtools')
Installing package into ‘/home/sodell/R/x86_64-pc-linux-gnu-library/3.5’
(as ‘lib’ is unspecified)
also installing the dependencies ‘openssl’, ‘gh’, ‘git2r’, ‘httr’, ‘usethis’

trying URL 'https://cloud.r-project.org/src/contrib/openssl_1.1.tar.gz'
Content type 'application/x-gzip' length 1192772 bytes (1.1 MB)
==================================================
downloaded 1.1 MB

trying URL 'https://cloud.r-project.org/src/contrib/gh_1.0.1.tar.gz'
Content type 'application/x-gzip' length 15513 bytes (15 KB)
==================================================
downloaded 15 KB

trying URL 'https://cloud.r-project.org/src/contrib/git2r_0.24.0.tar.gz'
Content type 'application/x-gzip' length 1151420 bytes (1.1 MB)
==================================================
downloaded 1.1 MB

trying URL 'https://cloud.r-project.org/src/contrib/httr_1.4.0.tar.gz'
Content type 'application/x-gzip' length 156356 bytes (152 KB)
==================================================
downloaded 152 KB

trying URL 'https://cloud.r-project.org/src/contrib/usethis_1.4.0.tar.gz'
Content type 'application/x-gzip' length 325703 bytes (318 KB)
==================================================
downloaded 318 KB

trying URL 'https://cloud.r-project.org/src/contrib/devtools_2.0.1.tar.gz'
Content type 'application/x-gzip' length 388953 bytes (379 KB)
==================================================
downloaded 379 KB

* installing *source* package ‘openssl’ ...
** package ‘openssl’ successfully unpacked and MD5 sums checked
Using PKG_CFLAGS=
------------------------- ANTICONF ERROR ---------------------------
Configuration failed because openssl was not found. Try installing:
 * deb: libssl-dev (Debian, Ubuntu, etc)
 * rpm: openssl-devel (Fedora, CentOS, RHEL)
```

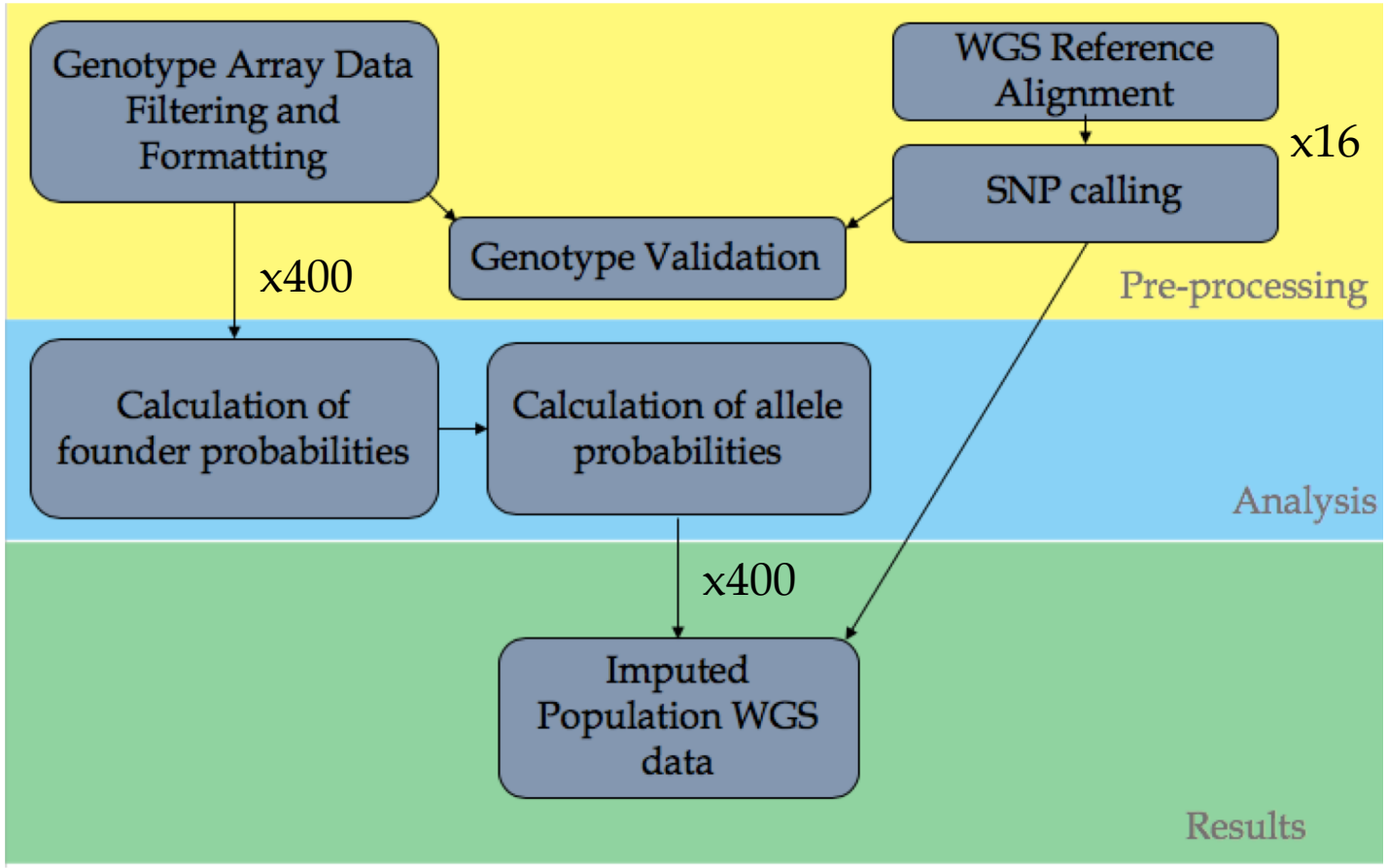
Building Workflows - **Conda**

A solution:

- General purpose package management system
- Allows you to create **environments** with required software tools and libraries
- Can specify specific versions of software to prevent updates from breaking your code

```
[sodell@c11-91:~$ python --version
Python 2.7.14
[sodell@c11-91:~$ module load conda3
[sodell@c11-91:~$ source activate sim_workshop
[(sim_workshop) sodell@c11-91:~$ python --version
Python 3.6.6
[(sim_workshop) sodell@c11-91:~$ conda list
# packages in environment at /share/apps/conda3/miniconda3/envs/sim_workshop:
#
# Name                    Version                   Build  Channel
asciitree                 0.3.3                      py_2    conda-forge
attrs                     18.2.0                     py_0    conda-forge
autopep8                  1.4.3                      py_0    conda-forge
backcall                  0.1.0                    py36_0    anaconda
bcolz                     1.2.1           py36hf8a1672_1    conda-forge
blas                      1.0                         mkl
blosc                     1.14.4                hfc679d8_0    conda-forge
bokeh                     1.0.1                  py36_1000    conda-forge
bzip2                     1.0.6                 h470a237_2    conda-forge
ca-certificates           2018.11.29            ha4d7672_0    conda-forge
certifi                   2018.11.29             py36_1000    conda-forge
click                     7.0                        py_0    conda-forge
```

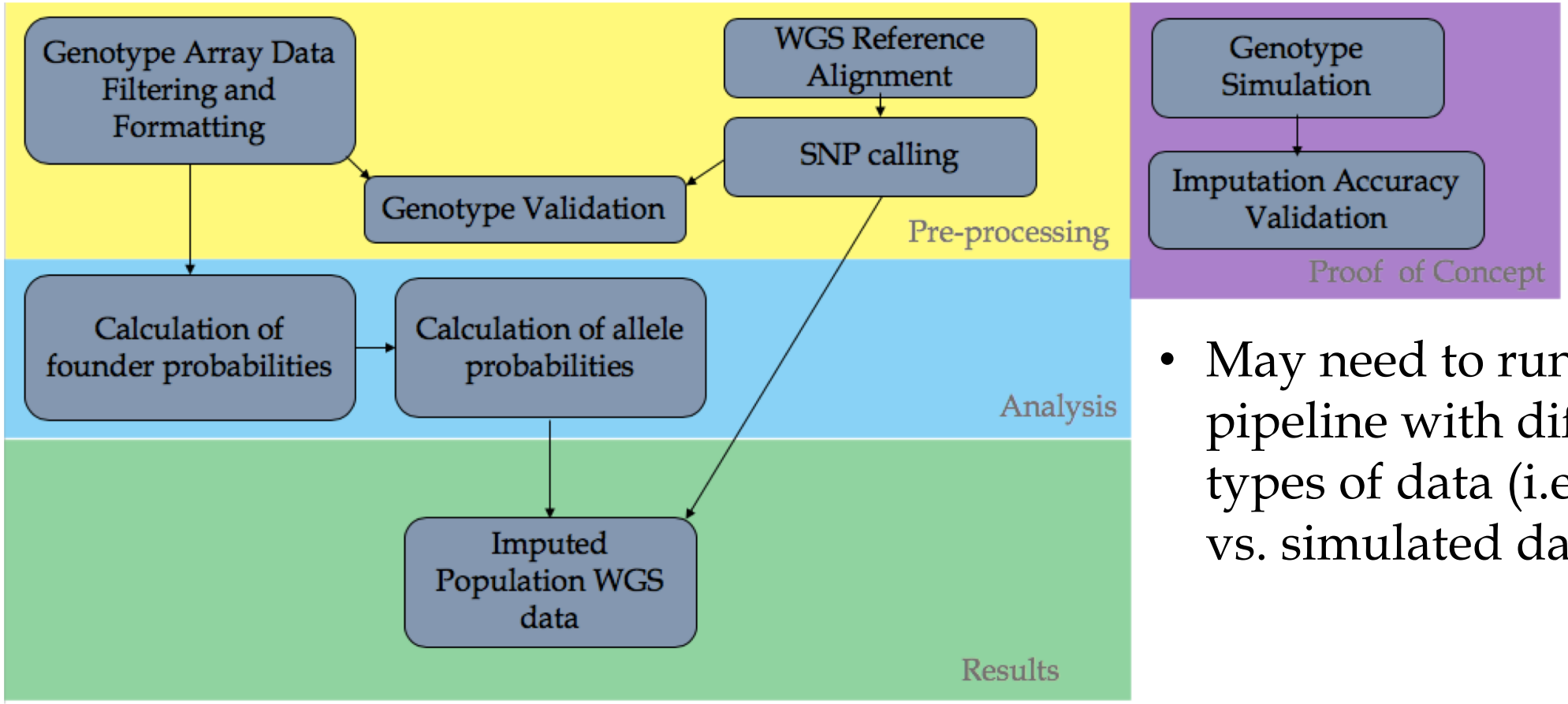
Building Workflows



The problems:

- May need to run the same thing many, many times
- Then have to take all of those results to use as input for further steps

Building Workflows



- May need to run the same pipeline with different types of data (i.e. real data vs. simulated data)

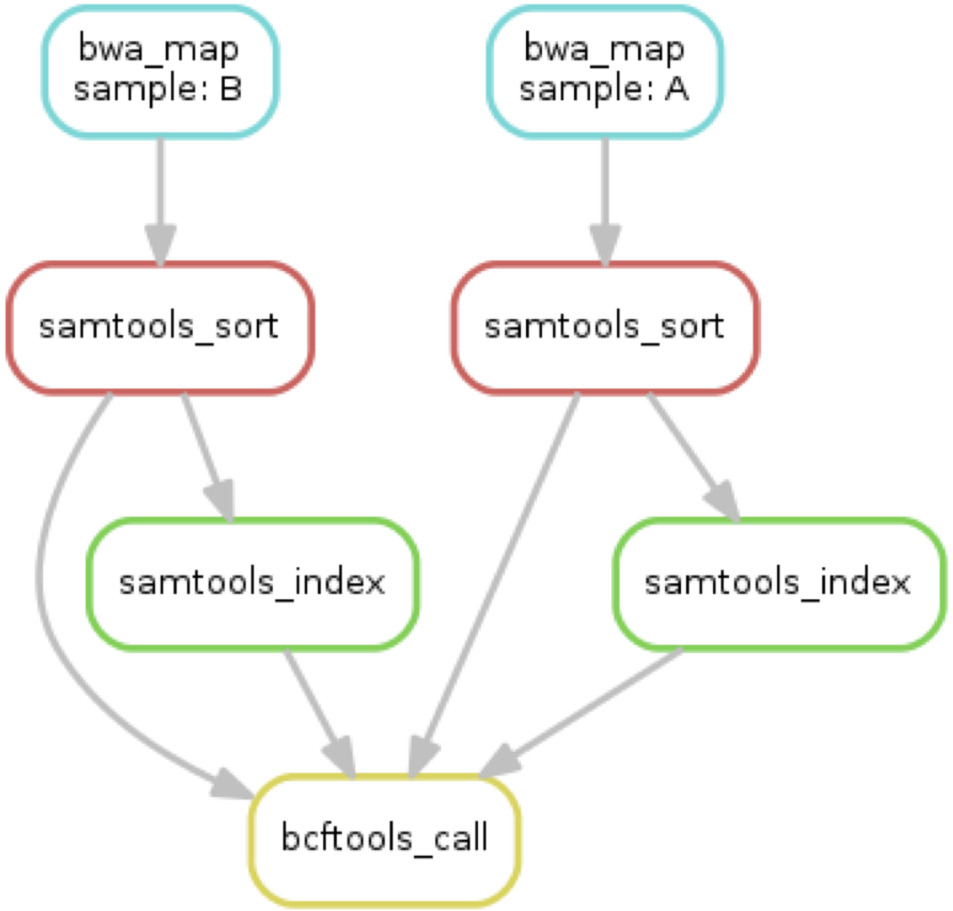
Building Workflows - **Snakemake**

```
rule bwa_map:
    input:
        "data/genome.fa",
        "data/samples/{sample}.fastq"
    output:
        "mapped_reads/{sample}.bam"
    shell:
        "bwa mem {input} | samtools view -Sb - > {output}"


rule samtools_sort:
    input:
        "mapped_reads/{sample}.bam"
    output:
        "sorted_reads/{sample}.bam"
    shell:
        "samtools sort -T sorted_reads/{wildcards.sample} "
        "-O bam {input} > {output}"
```

A solution:

- Workflow management system
- Breaks up workflow into small, generalizable steps
- Useful for automating pipelines

Building Workflows - **Snakemake**



A solution:

- Workflow management system
- Breaks up workflow into small, generalizable steps
- Useful for automating pipelines
- Can specify Conda environments
- You can create workflow diagrams of your pipeline!

Key Points

1. Thoughtful organization and naming of files and directories makes things findable later on.

2. Mixing code with comments and markdown (and soft-coding variables when necessary) makes code easier to run and interpret.

3. Documenting and diagraming workflows shows the bigger picture, allows generalization, and prevents breakdown.

Key Points

1. Thoughtful organization and naming of files and directories makes things findable later on.

2. Mixing code with comments and markdown (and soft-coding variables when necessary) makes code easier to run and interpret.

3. Documenting and diagraming workflows shows the bigger picture, allows generalization, and prevents breakdown.

Key Points

1. Thoughtful organization and naming of files and directories makes things findable later on.

2. Mixing code with comments and markdown (and soft-coding variables when necessary) makes code easier to run and interpret.

3. Documenting and diagraming workflows shows the bigger picture, allows generalization, and prevents breakdown.

Resources

Markdown: https://www.markdowntutorial.com/

Jupyter Notebooks: https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/

Intro to Git: https://nceas.github.io/oss-lessons/version-control/1-git-basics.html

Snakemake Tutorial: https://snakemake.readthedocs.io

Conda Environments: https://conda.io/docs/user-guide/tasks/manage-environments.html

References

Broman, K. & Woo, K. (2018) Data Organization in Spreadsheets, The American Statistician, 72:1, 2-10,DOI: 10.1080/00031305.2017.1375989

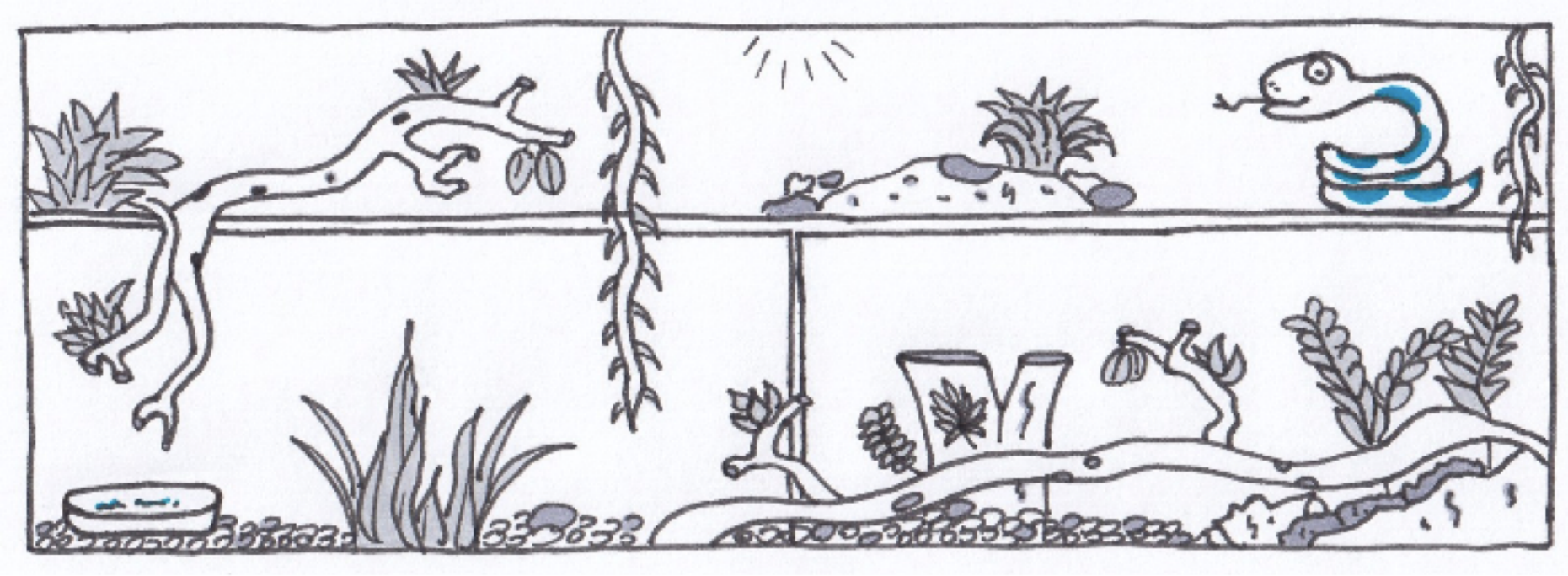
Noble WS (2009) A Quick Guide to Organizing Computational Biology Projects. PLOS Computational Biology 5(7): e1000424. https://doi.org/10.1371/journal.pcbi.1000424

Peng, R. (2011). Reproducible Research in Computational Science. Science, 334 (6060), 1226-1227. https://doi.org/10.1126/science.1213847

Stodden, V., McNutt, M., Bailey, D., Deelman, E., Gil, Y. et al. (2016) Enhancing reproducibility for computational methods. *Science*, 354 (6317), 1240-1241. https://doi.org/10.1126/science.aah6168

Garijo D., Kinnings S., Xie L., Xie L., Zhang Y., et al. (2013) Quantifying Reproducibility in Computational Biology: The Case of the Tuberculosis Drugome. PLOS ONE 8(11): e80278. https://doi.org/10.1371/journal.pone.0080278

Thank you!



This is how a perfect Python environment looks like ;)

https://bit.ly/2CfbXLu

Sarah Odell
sgodell@ucdavis.edu

Image Source: https://medium.freecodecamp.org/why-you-need-python-environments-and-how-to-manage-them-with-conda-85f155f4353c

Acknowledgments

Ross-Ibarra Lab

Jeffrey Ross-Ibarra

Asher Hudson

Runcie Lab

Daniel Runcie

RUNCIElab

The Plant Biology Graduate Group

Funding

Department of Plant Sciences Graduate Student Research Fellowship

UCDAVIS

DEPARTMENT OF PLANT SCIENCES

College of Agricultural and Environmental Sciences