Simulation Experiments in R

Helene Wagner, University of Toronto

Goals:

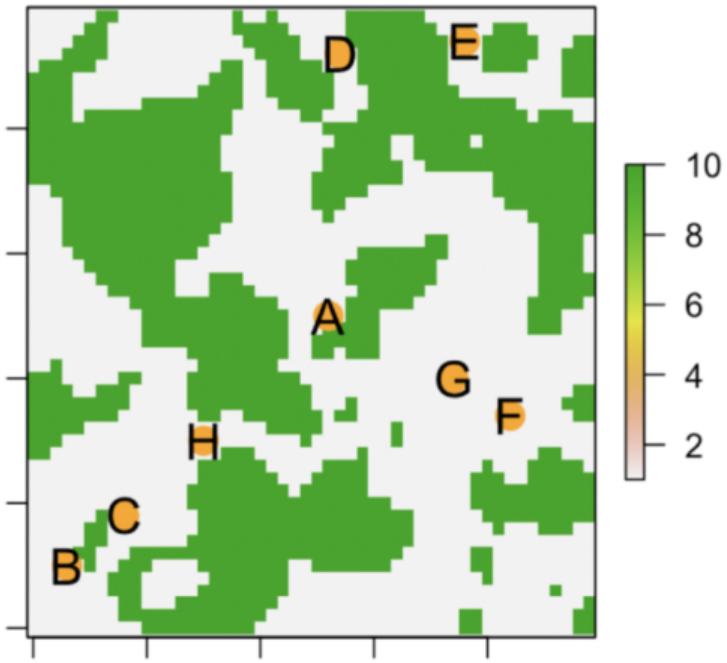
- Simulate a metapopulation on a resistance landscape
- Compare performance of partial Mantel test and Sunder

Methodological Challenges:

Video 1:

- Workflow of a simulation experiment
- Testing statistical methods with simulations
- Partial Mantel test vs. Sunder

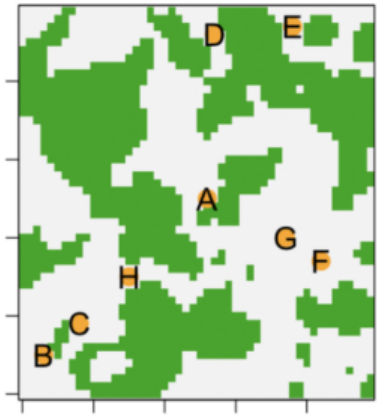
Video 2: Efficient R



Simulation Workflow

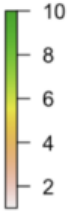
1. Initialize

- Landscape map — constant
- Populations (A - H) — constant
- Individuals (genotypes) — variable



Resistance:

Non-habitat: 10



Habitat: 1

Create random maps with 'secr':

- Habitat amount (A)
- Habitat aggregation (p)

2. Time step

- Demographic model
- Mating and reproduction
- Dispersal and recruitment

```
> str(para)
List of 15
 $ n.pops      : num 8
 $ n.ind       : num 100
 $ sex.ratio   : num 0.5
 $ n.cov       : num 3
 $ n.offspring: num 2
 $ mig.rate    : num 0.1
 $ disp.max    : num 50
 $ disp.rate   : num 0.05
 $ n.allels    : num 10
 $ n.loci      : num 20
 $ mut.rate    : num 0.001
```

3. Run a single simulation

- Initialize genotypes
- Run for many time steps
- Collect genotype data
- Summarize results

Fst: degree of differentiation

Decide: IBD or IBR?

4. Batch run simulations

- Replicate runs with same parameters
- Run scenarios across parameter space
- Store results and settings

Parameter space:

- time: # generations
- rep: # replicate sims

```
> para.space
  rep time
1   1    5
2   2    5
3   3    5
4   1   25
5   2   25
6   3   25
7   1   45
8   2   45
9   3   45
```

5. Synthesize results

- Extract summary data
- Visualize in parameter space
- Sensitivity analysis

Robust vs. sensitive

Partial Mantel Tests

IBD IBR

```
PopGenReport::wassermann(eucl.mat = eucl.mat, cost.mats = list(cost=cost.mat),
                         gen.mat = gen.mat, plot=F)$mantel.tab
```

	model <chr>		r <chr>	p <chr>
1	Gen ~cost | Euclidean	IBR | IBD	0.5366	0.041
2	Gen ~Euclidean | cost	IBD | IBR	-0.4753	0.983

Some issues with (partial) Mantel tests:

- Low statistical power?
- Inflated type I error rates if spatial autocorrelation?

Use simulations to test and compare methods!

Alternative with 'Sunder'

'Bedassle' (Bradburd et al. 2013), alternative implementation in 'Sunder' (Botta et al. 2014)

Run the analysis (parameter settings: http://www.nbi.dk/~botta/Sunder.html#overview)

IBD
IBR
Iterations

```
D.G <- as.matrix(dist(para$locs))
D.E <- cost.mat
nit <- 10^3   ## just for the example, should be much larger, e.g. 50000
output <- Sunder::MCMCCV(Array,D.G,D.E,
                         nit=nit,thinning=max(nit/10^3,1),
                         theta.max=c(10,10*max(D.G),10*max(D.E),1,0.9),
                         theta.init=c(1,2,1,1,0.01),
                         run=c(1,1,1), ud=c(0,1,1,0,0),
                         n.validation.set=dim(Array)[1]*dim(Array)[2]/10,
                         print.pct=FALSE)
```

```
print(output$mod.lik)
```

Likelihood

	IBD	IBR
G+E	G	E
-9050.244	-9058.499	-8974.353

```
> names(which.max(output$mod.lik))
[1] "E"
```

Testing Method Performance

Assessing error rates requires MANY replicate samples!

Resistance values

1	1	1	1	1
1	B	1	1	1
1	1	1	1	1
1	1	1	1	1
A	1	1	1	C

Type I error rate

- Simulate under null hypothesis
- Expect alpha % false positives

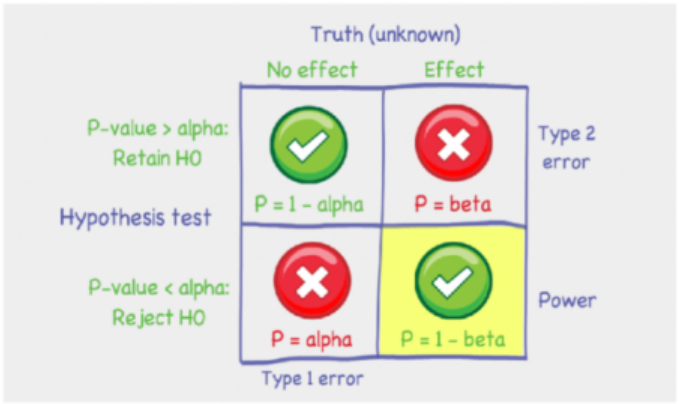
Statistical power to detect effect

- Simulate under alternative hypothesis
- Assess True Positive Rate (TPR)
- Larger effect size -> higher power
- Larger sample size -> higher power

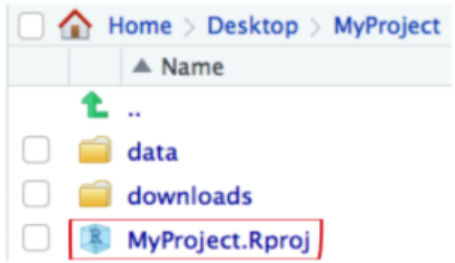
3	3	7	7	7
3	B	2	7	3
1	1	2	2	3
1	5	5	2	1
A	5	5	5	C

Compare power between methods!

Statistical Power



Where is my Stuff?



R has 3 homes !

`getwd( )`

Current working directory

"/Users/Helene/
Desktop/MyProject"

`Sys.getenv("HOME")`

User's home directory

"/Users/Helene"

`R.home( )`

Where R is installed

"/Library/Frameworks/
R.framework/Resources"

Default working directory?

- Console in regular R session: Varies! Use 'setwd( )'
- Console in R project: Project folder
- R Notebook: Notebook file location

"/Users/Helene/Desktop/MyProject/downloads"

Advice

- Always work in an R project
- Use path names relative to project folder
- Use 'here :: here( )' in R Notebooks

```
file.path ( here :: here( ), "data", "myFile.csv" )
```

"/Users/Helene/Desktop/MyProject/data/myFile.csv"

"./data/myFile.csv"

"/Users/Helene/Desktop/MyProject"

Input / Output

File System

List all files	`dir()`
Create directory	`dir.create()`
Download from web	`download.file()`
Unzip archive	`unzip()`
File size	`file.size()`
Remove file	`file.remove()`

R Workspace

`ls()`	List all objects
`object.size()`	Object size
`rm()`	Remove object

Text files: csv

base ::	`read.table()`	`write.table()`
base ::	`read.csv()`	`write.csv()`
readr ::	`read_csv()`	`write_csv()`
data.table ::	`fread()`	`fwrite()`

Binary files

base ::	`load()`	`save()`	.RData
base ::	`readRDS()`	`saveRDS()`	.rds
feather ::	`read_feather()`	`write_feather()`	.feather

Package 'rio':
Fast import / export for any file type

`import()`
`export()`

rds: Small, fast, flexible object format
feather: Compatibility with Python

Warning: check handling of text (character or factor?) and missing values!

Why is my Code Slow?

R was not designed to be fast!

1. Identify bottlenecks

Simple:
- Knit, monitor R Markdown pane
- Name each chunk in R Notebook
- Which chunks take a long time?

Advanced: profiling
- Convert .Rmd to .R: 'purl'
- Source script with 'source'
- Profile with 'lineprof' or 'profvis'
- Visualise time, memory use

Code	File	Time (ms)
▾ source		113850
▾ withVisible		113840
▾ eval		113840
▾ eval		113840
▸ PopGenReport::run.popgensim	Week8_vignette_A...	73970
▸ getSunder	Week8_vignette_A...	19060
▸ PopGenReport::wassermann	Week8_vignette_A...	7910
▸ mmod::pairwise_Gst_Nei	Week8_vignette_A...	3540
▸ secr::make.grid	Week8_vignette_A...	2910

2. Use faster functions

Simple:
- Vectorized: 'lapply' > 'for' loop
- Integrated: tidyverse > R base
- Optimized: CRAN task views

Advanced: benchmarking
- Package 'microbenchmark'
- Define each method as a function
- Compare speed: 'microbenchmark'
- Differences in precision, behavior?

```
Unit: milliseconds
                 expr      min       lq     mean   median
  import("gen.RData") 36.56224 39.62443 40.76476 40.21394
    import("gen.rds") 40.00667 40.12891 42.14960 41.90262
import("gen.feather") 45.26174 45.96569 46.91915 46.63800
    import("gen.csv") 73.59831 80.19661 90.62875 81.40790
```

3. Speed up your code

Simple:
- Preallocate result vectors
- Don't duplicate large objects
- Use binary data files

Advanced:
- Use 'data.table', 'bigmemory'
- Compile functions: 'cmpfun'
- Parallelize: use multiple cores
- Distribute: cluster computing

`mclapply()` `foreach()`

Bash R scripts 101

myBashFile.sh

Navigate the shell

- Default: same as ' Sys.genenv ("HOME") '
- In RStudio: project folder
- List folder content: ' ls '
- Move to folder with relative path:
  ' cd . /myFolder /subFolder '
- Move up one level: ' cd .. '

Execute a Bash R script

- Must change file permission
- Execute file
- Specify arguments

```
chmod +x myBashFile.sh
./ myBashFile.sh 5 0 1
```

Write a Bash R script ...

```
#!/bin/bash
R --slave << EOF

# Your R code:
myFunction <- function ( n, m, s)
{
   rnorm ( n, m, s )
}
args <- c( 5, 0, 1 )
myFunction ( args )

EOF
```

```
knitr :: purl ( "myNotebook.Rmd" )
creates file: myNotebook.R
```

... with arguments!

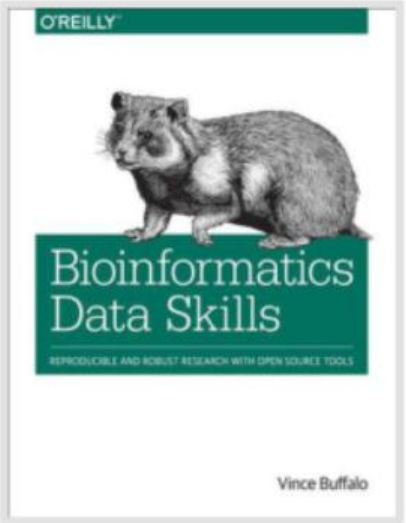
```
#!/bin/bash
R --slave --args $@ << EOF

# Your R code:
myFunction <- function ( n, m, s)
{
   rnorm ( n, m, s )
}
args <- as.numeric ( commandArgs ( ))
myFunction ( args )

EOF
```

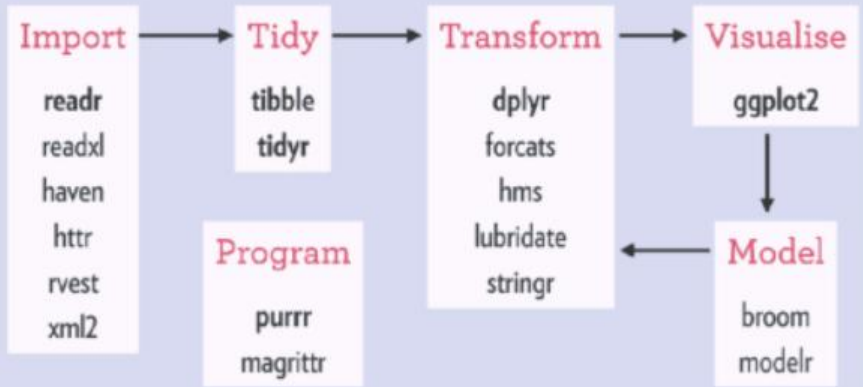
Remember what arguments to provide!
Arguments read as 'character' by default.

Further Reading

Books

Efficient R programming (Gillespie): https://csgillespie.github.io/efficientR
Advanced R (Wickham): http://adv-r.had.co.nz
R for Data Science (Wickham): http://r4ds.had.co.nz





Source: https://rviews.rstudio.com/2017/06/08/what-is-the-tidyverse/

Tidyverse

- Coherent system of packages for data manipulation, exploration and visualization
- Make data scientists more productive: workflow, communication, reproducible research

Blogs

https://www.r-bloggers.com/faster-higher-stonger-a-guide-to-speeding-up-r-code-for-busy-people/
https://www.r-bloggers.com/r-with-parallel-computing-from-user-perspectives/
https://datascienceplus.com/strategies-to-speedup-r-code/
https://support.rstudio.com/hc/en-us/articles/218221837-Profiling-with-RStudio
https://research.computing.yale.edu/sites/default/files/files/efficientR.pdf

Simulation Experiments in R

Helene Wagner, University of Toronto

Goals:

- Simulate a metapopulation on a resistance landscape
- Compare performance of partial Mantel test and Sunder

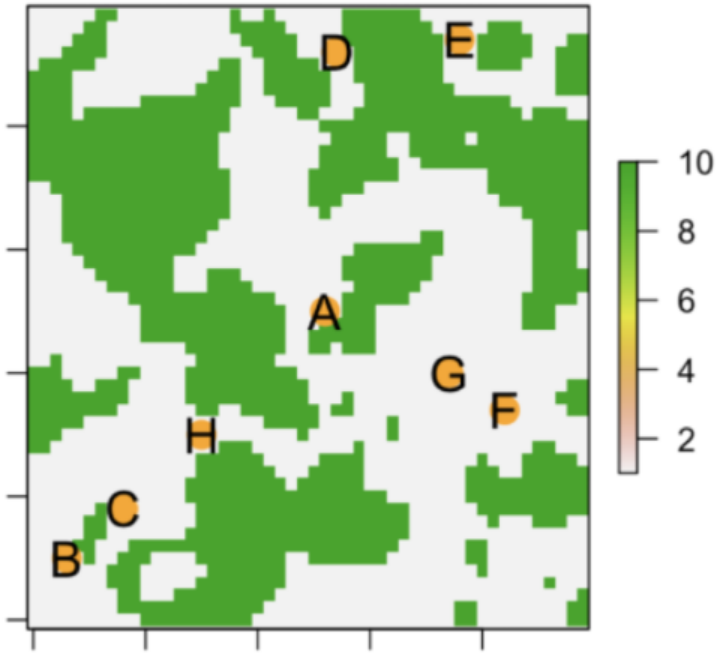
Methodological Challenges:

Video 1:

- Workflow of a simulation experiment
- Testing statistical methods with simulations
- Partial Mantel test vs. Sunder

Video 2: Efficient R

- Where is my stuff?
- Why is my code slow?



Interactive R tutorial:	generating data, string manipulation
Worked example:	landscape genetic simulation experiment
Bonus material:	file manipulation, benchmarking, Bash R script