# Type it to learn it

We strongly encourage you to type all of the code, rather than copying and pasting it from the resource below. Typing the code as you see it printed in the book will slow down your pace, allowing you to think about what are you typing and why are you using a certain command. You will experience how it feels to actually code – with all those forgotten closing brackets or semicolons that are so easy to miss. It might feel tedious and annoying at first, but you will learn to pay attention to these important details, making you more efficient in the long run.

CHAPTER 1

# Unix

## 1.1 What Is Unix?

## 1.2 Why Use Unix and the Shell?

## 1.3 Getting Started with Unix

### *1.3.1 Installation*

### *1.3.2 Directory Structure*

```
echo $HOME
```

## 1.4 Getting Started with the Shell

```
date
```

### *1.4.1 Invoking and Controlling Basic Unix Commands*

```
cal
```

```
cal 2020
```

```
cal -j
```

### *1.4.2 How to Get Help in Unix*

### *1.4.3 Navigating the Directory System*

```
cd ~/CSB/unix/sandbox
```

```
pwd
```

```
cd ~/CSB/python/data
cd ../../unix/data
cd -
```

```
cd Papers and reviews
cd Papers\ and\ reviews
cd "Papers and reviews"
cd Papers_and_reviews
```

## 1.5 Basic Unix Commands

### *1.5.1 Handling Directories and Files*

```
cp ~/CSB/unix/data/Buzzard2015_about.txt ~/CSB/unix/sandbox/
cp ../data/Buzzard2015_about.txt .
```

```
mv Buzzard2015_about2.txt ../data/
mv ../data/Buzzard2015_about2.txt ../data/Buzzard2015_about_new.txt
```

```
ls -l
touch new_file.txt
ls -l
```

```
rm -i new_file.txt
```

```
mkdir -p d1/d2/d3
rm -r d1
```

### *1.5.2 Viewing and Processing Text Files*

```
less Marra2014_data.fasta
```

```
cat Marra2014_about.txt Gesquiere2011_about.txt Buzzard2015_about.txt
```

```
wc Gesquiere2011_about.txt
wc -l Marra2014_about.txt
```

```
sort Gesquiere2011_data.csv
sort -n Gesquiere2011_data.csv
```

```
file Marra2014_about.txt
```

```
head -n 2 Gesquiere2011_data.csv
```

```
tail -n 2 Gesquiere2011_data.csv
tail -n +2 Gesquiere2011_data.csv
```

## 1.6 Advanced Unix Commands

### *1.6.1 Redirection and Pipes*

```
cd ~/CSB/unix/sandbox
```

```
echo "My first line" > test.txt
```

```
cat test.txt
```

```
echo "My second line" >> test.txt
cat test.txt
```

```
ls ../data/Saavedra2013 > filelist.txt
cat filelist.txt
```

```
wc -l filelist.txt
rm filelist.txt
```

```
ls  ../data/Saavedra2013 | wc -l
```

### *1.6.2 Selecting Columns Using `cut`*

```
cd ~/CSB/unix/data
head -n 1 Pacifici2013_data.csv
```

```
head -n 1 Pacifici2013_data.csv | cut -d ";" -f 1
head -n 1 Pacifici2013_data.csv | cut -d ";" -f 1-4
```

```
cut -d ";" -f 2 Pacifici2013_data.csv  | head -n 5
cut -d ";" -f 2,8 Pacifici2013_data.csv  | head -n 3
```

```
cut -d ";" -f 2 Pacifici2013_data.csv  | tail -n +2 | head -n 5
```

```
cut -d ";" -f 2 Pacifici2013_data.csv  | tail -n +2 | sort | uniq
```

### *1.6.3 Substituting Characters Using `tr`*

```
echo "aaaabbb" | tr "a" "b"
```

```
echo "123456789" | tr 1-5 0
```

```
echo "ACtGGcAaTT" | tr actg ACTG
```

```
echo "ACtGGcAaTT" | tr [:lower:] [:upper:]
```

```
echo "aabbccddee" | tr a-c 1-3
```

```
echo "aaaaabbbb" | tr -d a
```

```
echo "aaaaabbbb" | tr -s a
```

```
cat inputfile.csv | tr " " "\t" > outputfile.csv
tr " " "\t" < inputfile.csv > outputfile.csv
```

```
cd ../sandbox/
```

```
tail -n +2 ../data/Pacifici2013_data.csv
```

```
tail -n +2 ../data/Pacifici2013_data.csv | cut -d ";" -f 2-6
```

```
tail -n +2 ../data/Pacifici2013_data.csv | cut -d ";" -f 2-6 | tr ";" " "
```

```
tail -n +2 ../data/Pacifici2013_data.csv | cut -d ";" -f 2-6 | tr ";" " " |
    sort -r -n -k 6
```

```
tail -n +2 ../data/Pacifici2013_data.csv | cut -d ";" -f 2-6 | tr ";" " " |
    sort -r -n -k 6 > BodyM.csv
```

### *1.6.4 Wildcards*

```
cd ~/CSB/unix/data/miRNA
wc -l *.fasta
head -n 2 pp*
file *.???
```

### *1.6.5 Selecting Lines Using `grep`*

```
cd ~/CSB/unix/sandbox
wc -l BodyM.csv
```

```
grep Vombatidae BodyM.csv
```

```
grep -c Vombatidae BodyM.csv
```

```
grep Bos BodyM.csv
```

```
grep -w Bos BodyM.csv
```

```
grep -i Bos BodyM.csv
```

```
grep -B 2 -A 2 "Gorilla gorilla" BodyM.csv
```

```
grep -n "Gorilla gorilla" BodyM.csv
```

```
grep Gorilla BodyM.csv | grep -v gorilla
```

```
grep -w "Gorilla\|Pan" BodyM.csv
```

```
cd ~/CSB/unix
grep -r "Gorilla" data
```

### *1.6.6 Finding Files with `find`*

```
find ../data
```

```
find ../data | wc -l
```

```
find ../data -name "n30.txt"
```

```
find ../data -iname "*about*"
```

```
find ../data -name "*.txt" | wc -l
```

```
find ../data -maxdepth 1 -name "*.txt" | wc -l
```

```
find ../data -not -name "*about*" | wc -l
```

```
find ../data -type d
```

### *1.6.7 Permissions*

```
touch permissions.txt
ls -l
chmod u=rwx,g=rx,o=r permissions.txt
ls -l
chmod g-x,o+w permissions.txt
ls -l
```

```
mkdir -p test_dir/test_subdir
ls -l
sudo chown -R sallesina test_dir/
ls -l
```

## 1.7 Basic Scripting

```
touch ExtractBodyM.sh
```

```
gedit ExtractBodyM.sh &
```

```
open ExtractBodyM.sh &
```

```
tail -n +2 ../data/Pacifici2013_data.csv | cut -d ";" -f 2-6 | tr ";" " " |
    sort -r -n -k 6 > BodyM.csv
```

```
bash ExtractBodyM.sh
```

```
tail -n +2 ../data/Pacifici2013_data.csv | cut -d ";" -f 2-6 | tr ";" " " |
    sort -r -n -k 6 > BodyM.csv
```

```
bash ExtractBodyM.sh ../data/Pacifici2013_data.csv BodyM.csv
```

```
chmod +rx ExtractBodyM.sh
```

```
tail -n +2 $1 | cut -d ";" -f 2-6 | tr ";" " " | sort -r -n -k 6 > $2
```

```
./ExtractBodyM.sh ../data/Pacifici2013_data.csv BodyM.csv
```

```
tail -n +2 $1 > $1.tmp1
cut -d ";" -f 2-6 $1.tmp1  > $1.tmp2
tr ";" " " < $1.tmp2 > $1.tmp3
sort -r -n -k 6 $1.tmp3 > $2
rm $1.tmp*
```

## 1.8 Simple **for** Loops

```
cd ~/CSB/unix/data/miRNA
ls
```

```
for file in ggo_miR.fasta hsa_miR.fasta
```

```
for file in *.fasta
```

```
for miR in miR-208a miR-564 miR-3170
```

```
head -n 5 miR-564.fasta
```

## 1.9 Tips, Tricks, and Going beyond the Basics

### *1.9.1 Setting a **PATH** in **.bash_profile***

```
echo $PATH
```

```
export PATH=$PATH:[PATH TO PROGRAM]
```

```
which grep
```

### *1.9.2 Line Terminators*

### *1.9.3 Miscellaneous Commands*

## 1.10 Exercises

### *1.10.1 Next Generation Sequencing Data*

### *1.10.2 Hormone Levels in Baboons*

### *1.10.3 Plant–Pollinator Networks*

```
bash netsize.sh ../data/Saavedra2013/n1.txt
```

```
bash netsize_all.sh
```

### *1.10.4 Data Explorer*

```
bash explore.sh ../data/Buzzard2015_data.csv 7
```

## 1.11 References and Reading

### *Books*

### *Online Resources*

CHAPTER 2

# Version Control

## 2.1 What Is Version Control?

## 2.2 Why Use Version Control?

## 2.3 Getting Started with Git

### *2.3.1 Installing Git*

### *2.3.2 Configuring Git after Installation*

```
git config --global user.name "Charles Darwin"
git config --global user.email crdarwin@royalsociety.org
```

```
git config --global core.editor gedit
```

```
git config --global color.ui true
```

```
git config --list
```

### *2.3.3 How to Get Help in Git*

```
git help
```

```
man git
```

## 2.4 Everyday Git

### *2.4.1 Workflow*

```
cd ~/CSB/git/sandbox
mkdir originspecies
cd originspecies
git init
```

```
git status
```

```
touch origin.txt
```

```
echo "An Abstract of an Essay on the Origin of Species and Varieties
   Through Natural Selection" > origin.txt
```

```
cat origin.txt
```

```
git add origin.txt
```

```
git status
```

```
git commit -m "started the book"
```

```
git log
```

```
echo "On the Origin of Species, by Means of Natural Selection, or the
   Preservation of Favoured Races in the Struggle for Life" > origin.txt
```

```
git status
```

```
git add .
git commit -m "Changed the title as suggested by Murray"
```

```
git log
```

```
mkdir newproject
cd newproject
git init # initialize repository
git status
git add --all
git commit -m "my descriptive message"
```

### *2.4.2 Showing Changes*

```
git diff
```

```
git diff
```

### *2.4.3 Ignoring Files and Directories*

```
cat .gitignore
```

### *2.4.4 Moving and Removing Files*

```
git rm filetorem.txt
git rm *.txt
git mv myoldname.txt mynewname.csv
```

### *2.4.5 Troubleshooting Git*

```
git add forgottenfile.txt
git add fixedcode.py
git commit --amend
```

```
git reset HEAD filetounstage.py
```

```
git checkout filetoreset.txt
```

## 2.5 Remote Repositories

```
cd ~
git clone https://github.com/CSB-book/CSB.git
```

```
git pull
git add --all
git commit -m "A meaningful message"
git push
```

```
git stash
git status
git pull
git stash apply
```

## 2.6 Branching and Merging

```
less ../data/create_repository.sh
```

```
./../data/create_repository.sh
```

```
cd branching_example
git log --oneline --decorate
```

```
git branch fastercode
git log --oneline --decorate
```

```
git branch
```

```
git checkout fastercode
git branch
```

```
echo "Yeah, faster code" >> code.txt
```

```
git add code.txt
git commit -m "Managed to make code faster"
```

```
git log --oneline --decorate
```

```
git checkout master
echo "Marra et al. 2014" > references.txt
git add references.txt
git commit -m "Fixed the references"
```

```
git log --oneline --decorate --all --graph
```

```
git checkout fastercode
echo "# My documentation" >> code.txt
git add code.txt
git commit -m "Added comments to the code"
```

```
git log --oneline --decorate --all --graph
```

```
git checkout master
git merge fastercode -m "Much faster version of code"
```

```
git log --oneline --decorate --all --graph
```

```
git branch -d fastercode
git log --oneline --decorate --all --graph
```

```
git branch mybranch
git branch
git checkout mybranch
git add --all
git commit -m "my message"
git checkout master
git merge mybranch -m "message for merge"
git -d mybranch
```

```
git log --oneline --decorate --all --graph
```

```
git checkout bc4831a
```

```
git checkout master
```

```
git merge -m "We need to merge"
```

## 2.7 Contributing to Public Repositories

```
git clone https://github.com/YourName/CSB.git
cd CSB
git checkout master
git branch better_solution
git checkout better_solution
git add --all
git commit -m "Provide a detailed description of your changes"
git push origin better_solution
```

## 2.8 References and Reading

***Books and Tutorials***

***Graphical User Interface***

***Scientific Articles***

CHAPTER 3

# Basic Programming

## 3.1 Why Programming?

## 3.2 Choosing a Programming Language

## 3.3 Getting Started with Python

### *3.3.1 Installing Python and Jupyter*

### *3.3.2 How to Get Help in Python*

### *3.3.3 Simple Calculations with Basic Data Types*

```
2 + 2
2 * 2
3 / 2
3 // 2
2 > 3
2 == 2
2 != 2
"my string"
```

```
"""The tree's height is 6'2"."""
```

```
2 * 3 ** 3
```

```
2 * (3 ** 3)
(2 * 3) ** 3
```

```
15 % 7
```

### *3.3.4 Variable Assignment*

```
x = 5
x
```

```
who
```

```
x + 3
y = 8
x + y
```

```
x = "The cell grew"
x + " and is now larger"
```

```
x + y
```

```
x
y
x + " " + str(y) + " nm"
z = "88"
x + z
y + int(z)
```

```
x = 2
type(x)
x = "two"
type(x)
```

### *3.3.5 Built-In Functions*

```
s = "a long string"
len(s)
```

```
abs(-3.14)
pow(3, 6)
print(s)
round(3.1415926535, 3)
help(round)
```

### *3.3.6 Strings*

```
astring = "ATGCATG"
len(astring)
```

```
astring.
```

```
help(astring.find)
```

```
astring.replace("T", "U")
astring.find("C")
astring.count("G")
newstring = " Mus musculus "
newstring.split()
newstring.split("u")
newstring.strip()
```

```
"atgc".upper()
"TGCA".lower()
```

```
genus = "Rattus"
species = "norvegicus"
genus + " " + species
```

```
human = ["Homo", "sapiens"]
" ".join(human)
"->".join(["one", "leads", "2", "the", "other"])
```

```
s = "ATGC"
print(s)
s.print()
```

## 3.4 Data Structures

### *3.4.1 Lists*

```
new_list = []
my_list = [3, 2.44, "green", True]
a = list("0123456789")
a
```

```
my_list[1]
my_list[0]
```

```
my_list[4]
list index out of range
```

```
my_list[0] = "blue"
my_list
```

```
my_list
my_list[0:1]
my_list[1:3]
my_list[:]
my_list[:3]
my_list[3:]
```

```
my_list[-2]
```

```
my_list.append(25)
my_list
```

```
new_list = my_list.copy()
new_list
```

```
my_list.clear()
my_list
```

```
seq = list("TKAAVVNFT")
seq.count("V")
```

```
seq.index("V")
```

```
seq2 = seq.pop()
seq
seq2
```

```
a = [1, 5, 2, 42, 14, 132]
a.sort()
a
```

```
a.reverse()
a
```

```
del(a[2:3])
a
```

### *3.4.2 Dictionaries*

```
my_dict = {}
my_dict = {"a": "test", "b": 3.14, "c": [1, 2, 3, 4]}
my_dict
GenomeSize = {"Homo sapiens": 3200.0, "Escherichia coli": 4.6, "Arabidopsis
    thaliana": 157.0}
GenomeSize
GenomeSize["Arabidopsis thaliana"]
GenomeSize["Saccharomyces cerevisiae"] = 12.1
GenomeSize
GenomeSize["Escherichia coli"] = 4.6
GenomeSize
GenomeSize["Homo sapiens"] = 3201.1
GenomeSize
```

```
GS = GenomeSize.copy()
GS
```

```
GenomeSize.clear()
GenomeSize
```

```
GS.get("Mus musculus", -10)
```

```
GS.keys()
```

```
GS.values()
```

```
GS.pop("Homo sapiens")
GS
```

```
D1 = {"a": 1, "b": 2, "c": 3}
D2 = {"a": 2, "d": 4, "e": 5}
D1.update(D2)
D1
```

### *3.4.3 Tuples*

```
my_tuple = (1, "two", 3)
my_tuple[0]
my_tuple[0] = 33
```

```
tt = (1, 1, 1, 1, 2, 2, 4)
tt.count(1)
tt.index(2)
```

```
D3 = {("trial", 62): 4829}
```

### *3.4.4 Sets*

```
a = [5, 6, 7, 7, 7, 8, 9, 9]
b = set(a)
b
c = {3, 4, 5, 6}
b & c
b | c
b ^ c
```

```
s1 = {1, 2, 3, 4}
s2 = {4, 5, 6}
s1.intersection(s2)
s1.union(s2)
s1.symmetric_difference(s2)
s1.difference(s2)
s1.issubset(s2)
s1.issuperset(s2)
s1.issubset(s1.union(s2))
```

### *Summary of Data Structures*

```
type((1, 2))
type([1, 2])
type({1, 2})
type({1: "a", 2: "b"})
```

```
one = (1, 2, "tuple")
two = [3, 4, "list"]
three = {5: ["value1"], 6: ["value2"]}
container = [one, two, three]
container
container[2][5].append("value3")
container
```

## 3.5 Common, General Functions

```
max(a)
max(1.2, 3.71, 1.15)
max("scientific computing")
min("scientific computing")
```

```
sum(a)
sum(set([1, 1, 2, 3, 5, 8]))
```

```
"s" in "string"
36 not in [1, 2, 36]
(1, 2) in [(1, 3), (1, 2), 1000, 'aaa']
"z" in {"a": 1, "b": 2, "c": 3}
"c" in {"a": 1, "b": 2, "c": 3}
```

## 3.6 The Flow of a Program

### *3.6.1 Conditional Branching*

```
x = 4
if x % 2 == 0:
    print("Divisible by 2")
```

```
x = 4
if x % 2 == 0:
    print("Divisible by 2")
else:
    print("Not divisible by 2")
```

```
x = 17
if x % 2 == 0:
    print("Divisible by 2")
elif x % 3 == 0:
    print("Divisible by 3")
elif x % 5 == 0:
    print("Divisible by 5")
elif x % 7 == 0:
    print("Divisible by 7")
else:
    print("Not divisible by 2, 3, 5, 7")
```

```
if condition_is_true:
   execute_commands
elif other_condition_is_true:
   other_commands
else:
   commands_to_run_if_none_is_true
```

### *3.6.2 Looping*

```
x = 0
while x < 100:
    print(x)
    x = x + 1
```

```
a = 1
b = 1
c = 0
while c < 10000:
    c = a + b
    a = b
    b = c
    print(c)
```

```
a = True
while a:
    print("Infinite loop")
```

```
x = 15000
while x < 19000:
    if x % 19 == 0:
        print(str(x)  + " is divisible by 19")
        break
    x = x + 1
```

```
x = 0
found = 0
while found < 100:
    x = x + 1
    if x % 2 == 1:
        continue
    print(x)
    found = found + 1
```

```
z = [1, 5, "mystring", True]
for x in z:
    print(x)
```

```
my_string = "a given string"
for character in my_string:
    print(character)
```

```
z = {0: "a", 1: "b", 2: "c"}
z.items()
for (key, val) in z.items():
    print(key, "->", val)
```

```
list(range(10))
list(range(1, 5))
list(range(0, 10, 3))
```

```
for x in range(10):
    print(x ** 2)
```

```
list(enumerate(my_string))
```

```
for k, x in enumerate(my_string):
    print(k, x)
```

```
z = [1, 5, "mystring", True]
for element, value in enumerate(z):
    print("element: " + str(element) + " value: " + str(value))
```

```
a = [1, 2, 5, 14, 42, 132]
b = [x ** 2 for x in a]
print(b)
```

```
for i in range(3, 17):
    print("hello")
```

```
for j in range(12):
    if j % 3 == 0:
        print("hello")
```

```
for j in range(15):
     if j % 5 == 3:
        print("hello")
     elif j % 4 == 3:
        print("hello")
```

```
z = 0
while z != 15:
    print("hello")
    z = z + 3
```

```
z = 12
while z < 100:
    if z == 31:
        for k in range(7):
            print("hello")
    elif z == 18:
        print("hello")
    z = z + 1
```

```
for i in range(10):
    if i > 5:
        break
    print("hello")
```

```
z = 0
while z < 25:
    z = z + 1
    if z % 2 == 1:
        continue
    print("hello")
```

## 3.7 Working with Files

### *3.7.1 Text Files*

```
f = open("mytextfile.txt", "w")
```

```
f.name
f.mode
f.encoding
```

```
f.write(s + "\n")
f.writelines(["A\n", "B\n", "C\n"])
```

```
f.close()
```

```
with open("myfile.txt", "r") as f:
    for my_line in f:
        print(my_line)
```

```
inputfile = "test1.txt"
outputfile = "test2.txt"
with open(inputfile, "r") as fr:
    with open(outputfile, "w") as fw:
        for line in fr:
            fw.write(line)
```

```
f.seek(0)
```

```
f.next()
```

### *3.7.2 Character-Delimited Files*

```
with open("../data/Dalziel2016_data.csv") as f:
    for i, line in enumerate(f):
        print(line.strip())
        if i > 2:
           break
```

```
import csv
with open("../data/Dalziel2016_data.csv") as f:
    reader = csv.DictReader(f)
    for i, row in enumerate(reader):
        print(dict(row))
        if i > 2:
            break
```

```
with open("../data/Dalziel2016_data.csv") as fr:
    reader = csv.DictReader(fr)
    header = reader.fieldnames
    with open("Dalziel2016_Washington.csv", "w") as fw:
        writer = csv.DictWriter(fw, fieldnames = header, delimiter = ",")
        for row in reader:
            if row["loc"] == "WASHINGTON":
                writer.writerow(row)
```

## 3.8 Exercises

### *3.8.1 Measles Time Series*

### *3.8.2 Red Queen in Fruit Flies*

## 3.9 References and Reading

### *Books*

### *Documentation and Tutorials*

CHAPTER 4

# Writing Good Code

## 4.1 Writing Code for Science

## 4.2 Modules and Program Structure

### *4.2.1 Writing Functions*

```
def GCcontent(dna):
    dna = dna.upper()
    numG = dna.count("G")
    numC = dna.count("C")
    numA = dna.count("A")
    numT = dna.count("T")
    return (numG + numC) / (numG + numC + numT + numA)
```

```
whos
```

```
GCcontent("AATTTCCCGGGAAA")
GCcontent("ATGCATGCATGC")
```

```
def print_dictionary(mydic):
    for k, v in mydic.items():
        print("key: ", k, " value: ", str(v))
def squared(start = 1, end = 10):
    results = []
    for i in range(start, end):
        r = i ** 2
        results.append(r)
    return results
```

```
whos
```

```
print_dictionary({"a": 3.4, "b": [1, 2, 3, 4], "c": "astring"})
```

```
squared(start = 3, end = 10)
squared(5)
squared(end = 3)
squared()
```

```
def foo1(x = 7):
    return x ** 0.5
```

```
def foo2(x = 3, y = 5):
    if x > y:
       return x
    return y
```

```
def foo3(x = 2, y = 0, z = 9):
    if x > y:
       tmp = y
       y = x
       x = tmp
    if y > z:
       tmp = z
       z = y
       y = tmp
    if x > y:
       tmp = y
       y = x
       x = tmp
    return [x, y, z]
```

```
def foo4(x = 6):
    result = 1
    for i in range(1, x + 1):
        result = result * i
    return result
```

```
def foo5(x = 1729):
    d = 2
    myfactors = []
    while x > 1:
       if x % d == 0:
          myfactors.append(d)
          x = x / d
       else:
          d = d + 1
    return myfactors
```

```
def foo6(x = 25):
    if x == 1:
       return 1
    return x * foo6(x - 1)
```

```
def foo7(x = 100):
   myp = [2]
   for i in range(3, x + 1):
        success = False
        for j in myp:
            if i % j == 0:
                success = True
                break
        if success == False:
            myp.append(i)
    return myp
```

### *4.2.2 Importing Packages and Modules*

### *4.2.3 Program Structure*

```
import scipy
def build_population(N, p):
    """The population consists of N individuals.
       Each individual has two chromosomes, containing
       allele "A" or "a", with probability p or 1-p,
       respectively.
       The population is a list of tuples.
    """
    population = []
    for i in range(N):
        allele1 = "A"
        if scipy.random.rand() > p:
            allele1 = "a"
        allele2 = "A"
        if scipy.random.rand() > p:
            allele2 = "a"
        population.append((allele1, allele2))
    return population
```

```
build_population(N = 10, p = 0.7)
```

```
def compute_frequencies(population):
    """ Count the genotypes.
        Returns a dictionary of genotypic frequencies.
    """
    AA = population.count(("A", "A"))
    Aa = population.count(("A", "a"))
    aA = population.count(("a", "A"))
    aa = population.count(("a", "a"))
    return({"AA": AA,
            "aa": aa,
            "Aa": Aa,
            "aA": aA})
```

```
my_pop = build_population(6, 0.5)
my_pop
compute_frequencies(my_pop)
```

```
def reproduce_population(population):
    """ Create new generation through reproduction
        For each of N new offspring,
        - choose the parents at random;
        - the offspring receives a chromosome from
            each of the parents.
    """
    new_generation = []
    N = len(population)
    for i in range(N):
        dad = scipy.random.randint(N)
        mom = scipy.random.randint(N)
        chr_mom = scipy.random.randint(2)
        offspring = (population[mom][chr_mom], population [dad][1 - chr_mom
            ])
        new_generation.append(offspring)
    return(new_generation)
```

```
reproduce_population(my_pop)
```

```
import drift
```

```
def simulate_drift(N, p):
    my_pop = drift.build_population(N, p)
    fixation = False
    num_generations = 0
    while fixation == False:
        genotype_counts = drift.compute_frequencies       (my_pop)
        if genotype_counts["AA"] == N or genotype_counts["aa"] == N:
            print("An allele reached fixation at generation",
                num_generations)
            print("The genotype counts are")
            print(genotype_counts)
            fixation == True
            break
        my_pop = drift.reproduce_population(my_pop)
        num_generations = num_generations + 1
```

```
simulate_drift(100, 0.5)
simulate_drift(100, 0.9)
```

```
import pickle
pickle.dump(my_pop, open("population.pickle", "wb"))
```

```
population = pickle.load(open("population.pickle", "rb"))
print(population)
```

## 4.3 Writing Style

```
def my_function_name(argument1,
                     argument2,
                     argument3,
                     argument4,
                     argument5):
    one_operation()
    another_operation()
    if condition:
        yet_another_operation()
    return my_value
for each_item in my_list:
    for each_element in my_item:
        do_something()
```

```
def my_function_name(argument1,argument2,argument3,argument4,argument5):
 one_operation()
 another_operation()
 if condition:
         yet_another_operation()
 return my_value
for each_item in my_list:
  for each_element in my_item:
            do_something()
```

```
import scipy
import re
import scipy,re
```

```
a = b ** c
z = [a, b, c, d]
n = n * (n - 1)
k = (a * b) + (c * d)
my_list = another_list[1:n]
a=b**c
z=[ a,b,c,d ]
n =n*(n-1)
k=a*b+c*d
my_list = another_list[1 : n]
```

## 4.4 Python from the Command Line

```
#!/usr/bin/python3
import sys
```

```
if __name__ == "__main__":
    N = int(sys.argv[1])
    p = float(sys.argv[2])
    simulate_drift(N, p)
```

```
chmod +rx simulate_drift.py
```

```
./simulate_drift.py 1000 0.1
```

## 4.5 Errors and Exceptions

### *4.5.1 Handling Exceptions*

```
x = 6
y = 2.0
y / x
x = 0
y / x
```

```
y = 16.0
x = 0.0
try:
    print(y / x)
except:
    print("Cannot divide by 0")
print("I'm done")
```

```
y = 16.0
x = 0.0
try:
    print(y / x)
except ZeroDivisionError:
    print("cannot divide by 0")
print("I'm done")
```

## 4.6 Debugging

```
%pdb
```

```
get_expected_sqrt_x(sample_size = 100)
```

```
get_expected_sqrt_x(distribution = "normal", par1 = 1, par2 = 0.5,
    sample_size = 10)
```

```
get_expected_sqrt_x("normal", 1, 0.5, 1000)
```

```
%pdb
get_expected_sqrt_x("normal", 1, 0.5, 1000)
```

```
total
sample_size
sqrt(4)
```

```
z
```

```
from numpy.random import normal
from numpy.random import uniform
from math import sqrt
def get_expected_sqrt_abs_x(distribution = "uniform",
                        par1 = 0,
                        par2 = 1,
                        sample_size = 10):
    """ Calculate the expectation of sqrt(|X|)
    where X is a random variable.
    X can be either uniform or normal,
    with parameters specified by the user;
    before taking the square root, we take the
    absolute value, to make sure it's positive.
    """
    total = 0.0
    for i in range(sample_size):
        if distribution == "uniform":
            z = uniform(par1, par2, 1)
        elif distribution == "normal":
            z = normal(par1, par2, 1)
        else:
            print("Unknown distribution. Quitting...")
            return None
        total = total + sqrt(abs(z))
    return total / sample_size
```

```
import scipy
import scipy.special
def compute_likelihood_binomial(p, successes, trials):
    """ Compute the likelihood function for the binomial
        model where p is the probability of success;
        successes is the number of observed successes
        out of trials
    """
    assert p >= 0, "Negative probability"
    assert p <= 1, "Probability > 1!"
    assert successes <= trials, "More successes than trials!"
    log_likelihood = successes * scipy.log(p)
        + (trials - successes) * scipy.log(1.0 - p)
    return scipy.exp(log_likelihood) * scipy.special.binom (trials,
       successes)
```

```
compute_likelihood_binomial(0.5, 5, 10)
compute_likelihood_binomial(-0.5, 5, 10)
```

```
import pickle
genetic_code = pickle.load(open("../data/genetic_code.pickle", "rb"))
test_mRNA = "AUGGAAUUCUCGCUCUGAAGGUAA"
def get_amino_acids(mRNA):
    i = 0
    aa_sequence = []
    while (i + 3) < len(mRNA):
        codon = mRNA[i:(i + 3)]
        aa = genetic_code[codon]
        if aa == "Stop":
            break
        else:
            aa_sequence.append(aa)
        i = i + 4
    return "".join(aa_sequence)
print(get_amino_acids(test_mRNA))
```

# 4.7 Unit Testing

## *4.7.1 Writing the Tests*

```
def CGcontent(DNA):
    """ Return proportion of CG in sequence.
        Assumes that the DNA is uppercase containing
        ONLY A T G C
    """
    CG = DNA.count("C") + DNA.count("G")
    CG = CG / len(DNA)
    return CG
```

```
python3 -i CGcont.py
```

```
CGcontent("AAAAAA")
CGcontent("AAATTT")
CGcontent("AAATTTCCCC")
CGcontent("AAATTTCCC")
```

```
def CGcontent(DNA):
    """ Return proportion of CG in sequence.
        Assumes that the DNA is uppercase containing
        ONLY A T G C
    ================================
    Unit testing with docstrings
    ================================
    Run the command in python3, (e.g., python3 -i CGcont.py) and copy the
       output below:
    >>> CGcontent("AAAAAA")
    0.0
    >>> CGcontent("AAATTT")
    0.0
    >>> CGcontent("AAATTTCCCC")
    0.4
    >>> CGcontent("AAATTTCCC")
    0.333...
    """
    CG = DNA.count("C") + DNA.count("G")
    CG = CG / len(DNA)
    return CG
```

### *4.7.2 Executing the Tests*

```
python3 -m doctest -v CGcont.py
```

```
if __name__ == "__main__":
    import doctest
    doctest.testmod()
```

### *4.7.3 Handling More Complex Tests*

```
"""
>>> myfunction(x)
{'a': 1, 'b': 2, 'c': 3}
"""
```

```
"""
>>> tmp = myfunction(x)
>>> tmp == {'a': 1, 'b': 2, 'c': 3}
True
"""
```

```
import scipy
def get_sample_mean(n):
    my_sample = scipy.random.normal(size = n)
    return scipy.mean(my_sample)
```

```
get_sample_mean(10)
get_sample_mean(10)
```

```
scipy.random.seed(123)
get_sample_mean(10)
scipy.random.seed(123)
get_sample_mean(10)
```

```
import scipy
def get_sample_mean(n):
    """ For testing, we want to make sure we
        set the seed of the random number generator:
        >>> scipy.random.seed(1)
        >>> get_sample_mean(10)
        -0.0971...
    """
    my_sample = scipy.random.normal(size = n)
    return scipy.mean(my_sample)
```

## 4.8 Profiling

```
%run -p simulate_drift.py 1000 0.1
```

```
def reproduce_population2(population):
    """ Create new generation through reproduction
        For each of N new offspring,
        - choose the parents at random;
        - the offspring receives a chromosome from
            each of the parents.
    """
    N = len(population)
    new_generation = [("")] * N
    rai = scipy.random.randint
    for i in range(N):
        dad = rai(N)
        mom = rai(N)
        chr_mom = rai(2)
        offspring = (population[mom][chr_mom], population [dad][1 - chr_mom
            ])
        new_generation[i] = offspring
    return(new_generation)
```

```
%run drift.py
mypop = build_population(500, 0.1)
%timeit -r 10 reproduce_population(mypop)
%timeit -r 10 reproduce_population2(mypop)
```

## 4.9 Beyond the Basics

### *4.9.1 Arithmetic of Data Structures*

```
a = [1, 2, 3]
b = [4, 5]
a + b
a = (1, 2)
b = (4, 6)
a + b
z1 = {1: "AAA", 2: "BBB"}
z2 = {3: "CCC", 4: "DDD"}
z1 + z2
```

```
"a" * 3
(1, 2, 4) * 2
[1, 3, 5] * 4
```

### *4.9.2 Mutable and Immutable Types*

```
a = [1, 2, 3]
a[0] = 1000
a
```

```
dd = {"a": 1}
dd.update({"b": 2})
dd
dd["a"] = dd.get("a", 0) + 1
dd
```

```
a
a.sort()
a
b = a.pop()
a
```

```
tt = (1, 2, 3)
tt + (4, 5)
tt
ss = "a string"
ss.upper()
ss
```

### *4.9.3 Copying Objects*

```
a = 15
b = a
a = 32
a
b
```

```
a = [1, 2, 3]
b = a
a.append(4)
a
b
```

```
a = [1, 2, 3]
b = a.copy()
a.append(4)
a
b
```

```
a = [[1, 2], [3, 4]]
b = a.copy()
a[0][1] = 10
a
b
import copy
a = [[1, 2], [3, 4]]
b = copy.deepcopy(a)
a[0][1] = 10
a
b
```

### *4.9.4 Variable Scope*

```
def changea(x):
    a = x
    print("New value for a:", a)
a = 51
print("Current value of a:", a)
changea(22)
print("After calling the function:", a)
```

```
def changea(x):
    global a
    a = x
    print("New value for a:", a)
a = 51
print("Current value of a:", a)
changea(22)
print("After calling the function:", a)
```

## 4.10 Exercises

### *4.10.1 Assortative Mating in Animals*

### *4.10.2 Human Intestinal Ecosystems*

## 4.11 References and Reading

### *Style Guides*

### *Integrated Development Environment*

### *Unit Testing*

### *Magic Commands*

### *Zen*

```
import this
```

CHAPTER 5

# Regular Expressions

## 5.1 What Are Regular Expressions?

## 5.2 Why Use Regular Expressions?

## 5.3 Regular Expressions in Python

### *5.3.1 The `re` Module in Python*

```
import re
```

```
my_string = "a given string"
m = re.search(r"given", my_string)
```

```
print(m)
% <_sre.SRE_Match object; span=(2, 7), match='given'>
```

```
m.group()
```

```
m = re.search(r"9", my_string)
print(m)
```

## 5.4 Building Regular Expressions

### *5.4.1 Literal Characters*

### *5.4.2 Metacharacters*

```
my_string = "a given string"
m = re.search(r"\s", my_string)
m.group()
```

```
m = re.search(r"\s\w\w\w\w", my_string)
m.group()
```

### *5.4.3 Sets*

```
my_string = "sunflowers are described on page 89"
m = re.search(r"[sS]\w\w", my_string)
m.group()
m = re.search(r"[0-9][0-9]", my_string)
m.group()
```

```
m = re.search(r"[^s-z]\w\w\w\w\w", my_string)
m.group()
```

### *5.4.4 Quantifiers*

```
m = re.search(r"\s\w{5}", my_string)
m.group()
```

```
re.search(r"[ACGT]+","A possible explanation is the motif ATTCGT.").group()
```

```
re.search(r"[ACGT]{3,}", "the motif ATTCGT").group()
```

```
my_string = "once upon a time"
re.search(r".*\s", my_string)
m.group()
```

```
re.search(r".*?\s", "once upon a time").group()
```

### *5.4.5 Anchors*

```
my_string = "ATATA"
m = re.search(r"^TATA", my_string)
print(m)
m = re.search(r"TATA$", my_string)
m.group()
```

### *5.4.6 Alternations*

```
my_string = "I found my cat!"
m = re.search(r"cat|mouse", my_string)
m.group()
```

### *5.4.7 Raw String Notation and Escaping Metacharacters*

```
print("my long \n line")
print(r"my long \n line")
```

```
my_string = r"my long \n line"
m = re.search(r"\\n", my_string)
m.group()
```

```
with open("../data/Marra2014_BLAST_data.txt") as f:
    counter = 0
    for line in f:
        m = re.search(r"\*", line)
        if m:
            counter += 1
            print(line)
```

```
with open("../data/Marra2014_BLAST_data.txt") as f:
    for line in f:
        m = re.search(r"mhc[\s\w*]+\*\w*", line)
        if m:
            print(m.group())
```

## 5.5 Functions of the **re** Module

```
for my_match in re.finditer(reg, target_string):
  print(my_match.group())
```

```
help(re)
```

```
import re
import pyfaidx
genes = pyfaidx.Fasta("../data/Ecoli.fasta")
```

```
records = list(genes.keys())
records
seq1 = genes[records[0]][:]
seq1.end
40]
```

```
m = re.search(r"GATC", seq1_str)
m.group()
m.start()
m.end()
```

```
m = re.findall(r"AAC[ATCG]{6}GTGC|GCAC[ATCG]{6}GTT", seq1_str)
if m:
    print("There are " + str(len(m)) + " matches")
    print("These are the matches: " + str(m))
```

```
hits = re.finditer(r"AAC[ATCG]{6}GTGC|GCAC[ATCG]{6}GTT", seq1_str)
for item in hits:
    print(item.start() + 1, item.group())
```

```
EcoKI = re.compile(r"AAC[ATCG]{6}GTGC|GCAC[ATCG]{6}GTT")
m = EcoKI.findall(seq1_str)
m
```

## 5.6 Groups in Regular Expressions

```
re.search(r"GT{2}", "ATGGTGTCCGTGTT").group()
re.search(r"(GT){2}", "ATGGTGTCCGTGTT").group()
```

```
with open("../data/16S_regex_pattern.txt") as f:
    regpat = f.readline()
print(regpat)
```

```
seq2_str = genes[records[1]][:].seq
amp = re.compile(regpat)
m = amp.search(seq2_str)
m.group(0)[:40]
m.group(1)
m.group(2)[:40]
len(m.group(2))
m.group(3)
```

## 5.7 Verbose Regular Expressions

```
pattern = r"""
        ^
        (\d{5})
        ([\s-]{1}
        \d{4})?
        $
        """
```

```
re.search(pattern, "60637", re.VERBOSE).group()
re.search(pattern, "60637 1503", re.VERBOSE).group()
re.search(pattern, "60637-1503", re.VERBOSE).group()
```

## 5.8 The Quest for the Perfect Regular Expression

## 5.9 Exercises

### *5.9.1 Bee Checklist*

### *5.9.2 A Map of* Science

```
import matplotlib.pyplot as plt
%matplotlib inline
plt.scatter(zip_long, zip_lat, s = zip_count, c = zip_count)
plt.colorbar()
plt.xlim(-125,-65)
plt.ylim(23, 50)
ard = dict(arrowstyle="->")
plt.annotate("Los Angeles", xy = (-118.25, 34.05),
    xytext = (-108.25, 34.05), arrowprops = ard)
plt.annotate("Palo Alto", xy = (-122.1381, 37.4292),
    xytext = (-112.1381, 37.4292), arrowprops= ard)
plt.annotate("Cambridge", xy = (-71.1106, 42.3736),
    xytext = (-73.1106, 48.3736), arrowprops= ard)
plt.annotate("Chicago", xy = (-87.6847, 41.8369),
    xytext = (-87.6847, 46.8369), arrowprops= ard)
plt.annotate("Seattle", xy = (-122.33, 47.61),
    xytext = (-116.33, 47.61), arrowprops= ard)
plt.annotate("Miami", xy = (-80.21, 25.7753),
    xytext = (-80.21, 30.7753), arrowprops= ard)
params = plt.gcf()
plSize = params.get_size_inches()
params.set_size_inches( (plSize[0] * 3, plSize[1] *   3) )
plt.show()
```

## 5.10 References and Reading

CHAPTER 6

# Scientific Computing

## 6.1 Programming for Science

### *6.1.1 Installing the Packages*

## 6.2 Scientific Programming with **NumPy** and **SciPy**

### *6.2.1 NumPy Arrays*

```
import numpy as np
```

```
a = np.arange(9)
a
```

```
al = list(range(9))
al
al * 2
a * 2
2 + a
2 + al
```

```
a.shape
a.ndim
a.dtype.name
a.size
```

```
a.sum()
a.mean()
a.std()
a.min()
a.max()
```

```
np.sqrt(a)
np.exp(a)
```

```
a1 = np.array([1, 2, 3, 4])
print(a1)
a1.dtype.name
a1 = np.array([1.0, 2.0, 3.0, 4.0])
print(a1)
a1.dtype.name
m = np.array([[1, 2], [3, 4]])
m
m = np.array([[1, 2], [3, 4]], dtype = float)
print(m)
m.dtype.name
```

```
m = np.zeros((3, 2), dtype = float)
m
m = np.ones((2, 3), dtype = complex)
m
a = np.arange(9)
a.reshape((3,3))
np.random.random((2, 3))
```

```
a
a[1]
a[:4]
a[-3:]
m = np.arange(16).reshape((4, 4))
m
m[0:3, 1:4]
m[1]
m[:, 1]
```

```
m.sum()
m.sum(axis = 0)
m.sum(axis = 1)
```

```
m =  np.loadtxt("my_matrix_file.csv", delimiter = ",")
```

```
import numpy as np
import skimage.io as io
%matplotlib inline
```

```
image = io.imread("../data/Kacsoh2013_Drosobrain.png")
io.imshow(image)
```

```
image.shape
```

```
red = image[:,:,0]
red.mean()
red.std()
red.min()
red.max()
green = image[:,:,1]
green.mean()
```

```
img_copy = image.copy()
img_copy[:, 480:500, 0] = 100
io.imshow(img_copy)
```

```
threshold = 100
mask = red > threshold
io.imshow(mask)
mask.sum()
```

```
mask2 = red * (red > threshold)
io.imshow(mask2)
mask2.sum()
```

### *6.2.2 Random Numbers and Distributions*

```
import numpy as np
np.random.random(2)
```

```
np.random.random_integers(5, size = 10)
np.random.random_integers(-5, -3, size = 4)
```

```
a = np.arange(4)
a
np.random.shuffle(a)
a
```

```
np.random.beta(1/2, 1/2, 4)
np.random.standard_normal(size = 3)
np.random.normal(10, 0.1, 4)
mus = [1, 3]
cov = [[1, 0.3], [0.4, 2]]
np.random.multivariate_normal(mus, cov, 3)
```

```
print(np.random.random(1))
print(np.random.random(1))
```

```
np.random.seed(10)
print(np.random.random(1))
print(np.random.random(1))
np.random.seed(10)
print(np.random.random(1))
print(np.random.random(1))
```

### *6.2.3 Linear Algebra*

```
import numpy as np
import scipy
import scipy.linalg
```

```
M = scipy.random.random((3, 3))
M
scipy.linalg.eigvals(M)
scipy.linalg.det(M)
scipy.linalg.inv(M)
np.dot(M, scipy.linalg.inv(M))
```

### *6.2.4 Integration and Differential Equations*

```
import numpy as np
import scipy.integrate
def integrand(x):
    return ((x ** 2 - x) ** 4) / (1 + x ** 2)
my_integral = scipy.integrate.quad(integrand, 0, 1)
my_integral
22 / 7 - my_integral[0]
```

```
import numpy as np
import scipy
import scipy.integrate
def Gompertz(x, t, alpha, K):
    dxdt = x * alpha * np.log(K / x)
    return dxdt
x0 = 0.1
alpha = 1 / 2
K = 5
ts = np.arange(0, 10, 0.05)
y = scipy.integrate.odeint(Gompertz, x0, ts, args = (alpha, K))
```

```
import matplotlib.pyplot as plt
plt.plot(ts, y[:,0])
plt.xlabel("Time")
plt.ylabel("x(t)")
plt.show()
```

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.plot(t, x[:, 0], label = "x1")
plt.plot(t, x[:, 1], label = "x2")
plt.plot(t, x[:, 2], label = "x3")
plt.legend(loc = "best")
plt.xlabel("t")
plt.grid()
plt.show()
```

### *6.2.5 Optimization*

```
import numpy as np
import scipy.optimize
def my_eq(Sinf, a, r):
    return np.exp(-Sinf / (a / r)) - Sinf
```

```
my_eq(0.6, 0.16, 1/13)
my_eq(0.7, 0.16, 1/13)
my_eq(0.71, 0.16, 1/13)
```

```
scipy.optimize.root(my_eq, 0.01, args=(0.16, 1 / 13))
```

## 6.3 Working with **pandas**

```
import pandas
import numpy as np
```

```
data = pandas.read_csv("../data/Dale2015_data.csv")
```

```
data.shape
```

```
data.columns
```

```
data["Sum_scores"] = data["Female_plumage_score"] + data["Male_plumage_score
    "]
```

```
data["Study"] = 1
data["Rnd"] = np.random.random(data.shape[0])
```

```
del(data["Sum_scores"])
data.drop(["Rnd", "Study"], axis = 1, inplace = True)
```

```
data["Scientific_name"][:3]
data.Scientific_name[:3]
```

```
data.loc[:3, ["Scientific_name", "English_name"]]
data.loc[data.Scientific_name == "Zosterops mouroniensis"]
data.iloc[2, 1]
```

```
data[data.English_name.str.contains("Bowerbird")]["Scientific_name"][:3]
```

```
high_male_score = data[data["Male_plumage_score"] > 65]
```

```
highly_dimorphic = data[(data.Male_plumage_score > 70) & (
    data.Female_plumage_score < 40)]
```

```
high_male_score["Qualitative_score"] = "High"
```

```
high_male_score = data[data["Male_plumage_score"] >  65].copy()
high_male_score["Qualitative_score"] = "High"
```

```
data.Male_plumage_score.mean()
data.Male_plumage_score.median()
data.Male_plumage_score.std()
```

```
%matplotlib inline
```

```
data.Male_plumage_score.hist()
```

```
data.plot.scatter(x = "Male_plumage_score", y = "Female_plumage_score")
```

```
data[["Male_plumage_score", "Female_plumage_score"]].plot.box()
```

## 6.4 Biopython

### *6.4.1 Retrieving Sequences from NCBI*

```
from Bio import Entrez
Entrez.email = "me@bigu.edu"
handle = Entrez.esearch(
    db = "nuccore",
    term = ("Uropsilus investigator[Organism]"))
```

```
record = Entrez.read(handle)
handle.close()
record.keys()
dict_keys(['Count', 'RetMax', 'RetStart', 'IdList', 'TranslationSet', '
   TranslationStack', 'QueryTranslation'])
```

```
record["Count"]
id_list = record["IdList"]
print(id_list)
```

```
Entrez.email = "me@bigu.edu"
handle = Entrez.efetch(db = "nuccore",
                       rettype = "fasta",
                       retmode = "text",
                       id = id_list[:10])
out_handle = open("Uropsilus_seq.fasta", "w")
for line in handle:
    out_handle.write(line)
out_handle.close()
handle.close()
```

### *6.4.2 Input and Output of Sequence Data Using `SeqIO`*

```
from Bio import SeqIO
handle = open("Uropsilus_seq.fasta", "r")
for record in SeqIO.parse(handle, "fasta"):
    print(record.description)
    print(len(record))
handle.close()
```

```
import re
output_handle = open("Uropsilus_BMI1.fasta", "w")
for record in SeqIO.parse("Uropsilus_seq.fasta", "fasta"):
    if re.search("BMI1", record.description):
        print(record.id)
        short_seq = record[:100]
        SeqIO.write(short_seq, output_handle, "fasta")
output_handle.close()
```

### *6.4.3 Programmatic BLAST Search*

```
from Bio.Blast import NCBIWWW
handle = open("Uropsilus_BMI1.fasta", "r")
records = list(SeqIO.parse(handle, "fasta"))
print(records[3].id, " ", records[3].seq)
```

```
Entrez.email = "me@bigu.edu"
result_handle = NCBIWWW.qblast("blastn", "nt", records[3].seq)
save_file = open("my_blast.xml", "w")
save_file.write(result_handle.read())
save_file.close()
result_handle.close()
```

```
from Bio.Blast import NCBIXML
result_handle = open("my_blast.xml")
blast_records = NCBIXML.read(result_handle)
```

```
E_VALUE_THRESH = 0.04
for alignment in blast_records.alignments:
    for hsp in alignment.hsps:
        if hsp.expect < E_VALUE_THRESH and alignment.length > 3000:
            print("****Alignment****")
            print("sequence:", alignment.title)
            print("length:", alignment.length)
            print("E value:", hsp.expect)
            print(hsp.query[0:75] + "...")
            print(hsp.match[0:75] + "...")
            print(hsp.sbjct[0:75] + "...")
```

### *6.4.4 Querying PubMed for Scientific Literature Information*

```
Entrez.email = "me@bigu.edu"
handle = Entrez.esearch(
    db = "pubmed",
    term = ("spaetzle[Title/Abstract] AND Drosophila[ALL]"),
    usehistory = "y")
record = Entrez.read(handle)
handle.close()
record["Count"]
'13'
```

```
webenv = record["WebEnv"]
query_key = record["QueryKey"]
```

```
Entrez.email = "me@bigu.edu"
handle = Entrez.efetch(db = "pubmed",
                 rettype = "medline",
                 retmode = "text",
                 webenv = webenv,
                 query_key = query_key)
out_handle = open("Spaetzle_abstracts.txt", "w")
data = handle.read()
handle.close()
out_handle.write(data)
out_handle.close()
```

```
import re
with open("Spaetzle_abstracts.txt") as datafile:
    pubmed_input = datafile.read()
    pubmed_input = re.sub(r"\n\s{6}", " ", pubmed_input)
    for line in pubmed_input.split("\n"):
        if re.match("PMID", line):
            PMID = re.search(r"\d+", line).group()
        if re.match("AB", line):
            spaetzle = re.findall(r"([^.]*?Spaetzle[^.]*\.)", line)
            if spaetzle:
                print("PubMedID: ", PMID, " ", spaetzle)
```

## 6.5 Other Scientific Python Modules

## 6.6 Exercises

### *6.6.1 Lord of the Fruit Flies*

### *6.6.2 Number of Reviewers and Rejection Rate*

### *6.6.3 The Evolution of Cooperation*

## 6.7 References and Reading

### *`SciPy`*

### *`pandas`*

### *`Biopython`*

### *Other Packages*

CHAPTER 7

# Scientific Typesetting

7.1 What Is LaTeX?

7.2 Why Use LaTeX?

7.3 Installing LaTeX

7.4 The Structure of LaTeX Documents

```
\documentclass[12pt]{article}
\title{A simple \LaTeX{} document}
\author{Stefano Allesina, Madlen Wilmes}
```

### 7.4.1 Document Classes

### 7.4.2 LaTeX Packages

### 7.4.3 The Main Body

```
\documentclass[12pt]{article}
\title{A simple \LaTeX{} document}
\author{Stefano Allesina, Madlen Wilmes}
\date{}

\begin{document}
\maketitle

\begin{abstract}
   Every scientific article needs a concise, clearly
   written abstract that summarizes the findings.
   It should leave the reader with no choice but to read
   the entire paper and cite it as soon as possible.
\end{abstract}

\end{document}
```

```
pdflatex My_document.tex
pdflatex My_document.tex
```

### 7.4.4 Document Sections

```
\section{A section}
\subsection{A subsection}
\subsubsection{A subsubsection}
\paragraph{A paragraph has no numbering}
\subparagraph{A subparagraph has no numbering}
```

```
\chapter{My Chapter Heading}
```

## 7.5 Typesetting Text with LaTeX

### *7.5.1 Spaces, New Lines, and Special Characters*

### *7.5.2 Commands and Environments*

```
\section{Introduction}
Every scientific publication needs an introduction.

\section{Materials \& Methods}
The \textbf{Hardy--Weinberg equilibrium model}\footnote{Godfrey Hardy, an
   English mathematician, and Wilhelm Weinberg, a German physician,
   developed the concept independently.} constitutes the \textit{null model}
    of population genetics. It characterizes the distributions of \texttt{
   genotype frequencies} in populations that are \underline{not evolving}.
```

### *7.5.3 Typesetting Math*

```
We assume that $p$ is the frequency of the dominant allele ($A$) and  $q$ is
    the frequency of the recessive allele ($a$). Given that the sum of the
    frequencies of both alleles in a population in genetic equilibrium is 1,
    we can write
\[ p + q = 1, \]
hence
\[ (p + q)^2 = 1, \]
which resolves to
\begin{equation}
p^2 + 2pq + q^2 = 1.
\end{equation}
In this equation, $p^2$ is the predicted frequency of homozygous dominant (
    $AA$) individuals in a population, $2pq$ is the predicted frequency of
    heterozygous ($Aa$) individuals, and $q^2$ is the predicted frequency of
    homozygous recessive ($aa$) ones.
```

### *7.5.4 Comments*

### *7.5.5 Justification and Alignment*

### *7.5.6 Long Documents*

```
\documentclass{report}

\usepackage{amsmath}
\usepackage{graphicx}
\graphicspath{{Pictures/}}

\begin{document}

\title{\textbf{Your short or long thesis title}}
\author{Your Name}

\maketitle
\tableofcontents

\input{introduction}
\input{methods}
\input{results}
\input{discussion}

\end{document}
```

### *7.5.7 Typesetting Tables*

```
\section{Results}

\subsection{Genotype Frequencies}

We estimated genotype frequencies of 1612 individuals.

\begin{table}[h]
  \begin{center}
    \caption{Frequency distribution of observed phenotypes.}
    \vspace{.1in}
    \begin{tabular}{lc} \textbf{Phenotype} & \textbf{Frequency} \\
      \hline
      $AA$ & 1469 \\
      $Aa$ & 138 \\
      $aa$ & 5 \\
      \hline
      \textbf{Total} & \textbf{1612}\\
     \end{tabular}
   \end{center}
\end{table}
```

### *7.5.8 Typesetting Matrices*

### *7.5.9 Figures*

```
\subsection{Gel Electrophoresis}

We identified differences in allelic composition by gel electrophoresis.

\begin{figure}[h]
  \begin{center}
     \includegraphics{electrophoresis.png}
  \end{center}
 \caption{Gel electrophoresis of four individuals.}
\end{figure}
```

```
\subsection{Gel Electrophoresis}

We identified differences in allelic composition by gel electrophoresis.

\begin{figure}[h]
  \begin{center}
     \includegraphics[scale=0.8, trim={0cm 0cm 2.6cm 0cm},clip]{
        electrophoresis.png}
  \end{center}
 \caption{Gel electrophoresis of one individual. A molecular-weight size
    marker is shown to the left for comparison.}
\end{figure}
```

### *7.5.10 Labels and Cross-References*

```
\subsection{Gel Electrophoresis}\label{sec:electrophoresis}

We identified differences in allelic composition by gel
electrophoresis (figure~\ref{fig:electrophoresis}).

\begin{figure}[h]
  \begin{center}
     \includegraphics[scale=0.8, trim={0cm 0cm 2.6cm 0cm},clip]{
        electrophoresis.png}
  \end{center}
 \caption{Gel electrophoresis of one individual. A molecular-weight size
    marker is shown to the left for comparison.}\label{fig:electrophoresis}
\end{figure}

\section{Discussion}

Our experiment demonstrates differences in allelic composition
(section~\ref{sec:electrophoresis}).
```

### *7.5.11 Itemized and Numbered Lists*

### *7.5.12 Font Styles*

### *7.5.13 Bibliography*

```
\section{Materials \& Methods}
The \textbf{Hardy--Weinberg equilibrium model}\footnote{Godfrey Hardy, an
   English mathematician, and Wilhelm Weinberg, a German physician,
   developed the concept independently.} constitutes the \textit{null model}
    of population genetics. It characterizes the distributions of \texttt{
   genotype frequencies} in populations that are \underline{not evolving} \
   cite{Hardy1908, Weinberg1908}.
```

```
\bibliography{My_Document}
\bibliographystyle{plain}
```

```
pdflatex My_Document.tex
pdflatex My_Document.tex
bibtex My_Document
pdflatex My_Document.tex
pdflatex My_Document.tex
```

## 7.6 LaTeX Packages for Biologists

### *7.6.1 Sequence Alignments with LaTeX*

```
\begin{texshade}{PLA2.fasta}
\setends{1}{34..56}
\showsequencelogo{top}
\feature{top}{1}{39..54}{helix}{Helix II}
\showruler{bottom}{1}
\hidenumbering
\end{texshade}
```

### *7.6.2 Creating Chemical Structures with LaTeX*

## 7.7 Exercises

### *7.7.1 Typesetting Your Curriculum Vitae*

## 7.8 References and Reading

### *Books*

### *Online Resources*

### *Tutorials and Essays*

CHAPTER 8

# Statistical Computing

8.1 Why Statistical Computing?

8.2 What Is R?

8.3 Installing R and RStudio

8.4 Why Use R and RStudio?

8.5 Finding Help

8.6 Getting Started with R

```
1.7 * 2
12 / 5
2.1 ^ 5
log(10)
log10(10)
sqrt(9)
q()
```

```
5 > 3
5 == (10 / 2)
6 > 2 ^ 4
6 >= (2 * 3)
(5 > 3) & (7 < 5)
(5 > 3) | (7 < 5)
```

```
3 %in% 1:5
c(2, "good to be", TRUE)
2 %in% c(2, "good to be", TRUE)
1:8 %in% c(1, 3, 5)
```

## 8.7 Assignment and Data Types

```
x <- 5
x * 2
x <- 7
x * 2
```

```
x <- as.integer(5)
class(x)
x <- pi
class(x)
x <- 1 + 3i
class(x)
x <- (5 > 7)
class(x)
x <- "hello"
class(x)
is.numeric(x)
```

```
x <- 5
as.character(x)
as.logical(x)
y <- "07.123"
x < y
x < as.numeric(y)
```

# 8.8 Data Structures

## *8.8.1 Vectors*

```
x <- c(2, 3, 5, 27, 31, 13, 17, 19)
x
```

```
x[3]
x[8]
x[9]
```

```
x[1:3]
x[4:7]
x[c(1,3,5)]
```

```
length(x)
min(x)
max(x)
sum(x)
prod(x)
median(x)
mean(x)
var(x)
mean(x ^ 2) - mean(x) ^ 2
summary(x)
```

```
x <- 1:10
x
```

```
seq(from = 1, to = 5, by = 0.5)
```

```
rep("abc", 3)
rep(c(1,2,3), 3)
```

### *8.8.2 Matrices*

```
A <- matrix(c(1, 2, 3, 4), 2, 2)
A
A %*% A
solve(A)
A %*% solve(A)
diag(A)
B <- matrix(1, 3, 2)
B
B %*% t(B)
Z <- matrix(1:9, 3, 3)
Z
```

```
dim(B)
nrow(B)
ncol(B)
```

```
Z
Z[1, ]
Z[, 2]
Z[1:2, 2:3]
Z[c(1,3), c(1,3)]
```

```
sum(Z)
mean(Z)
```

Arrays

```
M <- array(1:24, dim = c(4, 3, 2))
M
dim(M)
```

```
M[,,1]
```

```
dim(M[,,1])
```

```
dim(M[,,1, drop = FALSE])
```

### *8.8.3 Lists*

```
mylist <- list(Names = c("a", "b", "c", "d"),
mylist
mylist[[1]]
mylist[[2]]
mylist$Names
mylist[["Names"]]
mylist[["Values"]][3]
```

### *8.8.4 Strings*

```
x <- "Sample-36"
strsplit(x, '-')
sprintf("%s contains %d nucleotide types", "DNA", 4)
substr(x, start = 8, stop = 9)
sub("36", "39", x)
paste(x, "is significantly smaller", sep = " ")
nchar(x)
toupper(x)
tolower(x)
```

### *8.8.5 Data Frames*

```
data(trees)
str(trees)
ncol(trees)
nrow(trees)
head(trees)
trees$Girth
trees$Height[1:5]
trees[1:3, ]
trees[1:3, ]$Volume
trees <- rbind(trees, c(13.25, 76, 30.17))
trees_double <- cbind(trees, trees)
colnames(trees) <- c("girth", "height", "volume")
```

## 8.9 Reading and Writing Data

```
read.csv("MyFile.csv")
read.csv("MyFile.csv", header = TRUE)
read.csv("MyFile.csv", sep = ';')
read.csv("MyFile.csv", skip = 5)
```

```
write.csv(MyDF, "MyFile.csv")
write.csv("MyFile.csv", append = TRUE)
read.csv("MyFile.csv", row.names = TRUE)
read.csv("MyFile.csv", col.names = FALSE)
```

```
ch6 <- read.table("../data/H938_Euro_chr6.geno", header = TRUE)
```

```
dim(ch6)
```

```
head(ch6)
```

```
tail(ch6)
```

## 8.10 Statistical Computing Using Scripts

### *8.10.1 Why Write a Script?*

### *8.10.2 Writing Good Code*

```
plotting.R
model_fitting.R
misc.R
stuff.r
load_me.R
```

```
my_variance
radius
body_mass
tmp5
foo
good
I
O
o
l
calculate_cv
read_fasta
f1
faster_version
this_works_well
```

```
x <- 5 * 7
y <- 7 * (x ^ 2)
m <- matrix(25, 5, 5)
z <- mean(x, na.rm == TRUE)
i <- t + 1
z <- (x * y) + (x2 * y2)
if (b == 5) {
  do(something)
} else {
  do(something_else)
}
z = "bad_assignment_style"
x<-5*7
y <- 7*x^2
m <- matrix(25 , 5 , 5)
```

# 8.11 The Flow of the Program

## *8.11.1 Branching*

```
z <- readline(prompt = "Enter a number: ")
```

```
z <- readline(prompt = "Enter a number: ")
z <- as.numeric(z)
```

```
z <- readline(prompt = "Enter a number: ")
z <- as.numeric(z)
if (z %% 2 == 0) {
  print(paste(z, "is even"))
} else {
  print(paste(z, "is odd"))
}
```

```
if (a condition is TRUE) {
  execute these commands
} else {
  execute these other commands [optional]
}
```

```
source("conditional.R")
source("conditional.R")
```

### *8.11.2 Loops*

```
myvec <- 1:10
for (i in myvec) {
  a <- i ^ 2
  print(a)
}
```

```
for (variable in list_or_vector) {
  execute these commands
}
```

```
i <- 1
while (i <= 10) {
  a <- i ^ 2
  print(a)
  i <- i + 1
}
```

```
while (condition is TRUE) {
  execute these commands
  update the condition
}
```

```
i <- 1
while (i <= 10) {
  if (i > 5) {
    break
  }
  a <- i ^ 2
  print(a)
  i <- i + 1
}
```

```
z <- seq(1, 1000, by = 3)
for (k in z) {
  if (k %% 4 == 0) {
     print(k)
  }
}
```

```
z <- readline(prompt = "Enter a number: ")
z <- as.numeric(z)
isthisspecial <- TRUE
i <- 2
while (i < z) {
  if (z %% i == 0) {
     isthisspecial <- FALSE
     break
  }
  i <- i + 1
}
if (isthisspecial == TRUE) {
  print(z)
}
```

## 8.12 Functions

```
my_func_name <- function([optional arguments]) {
  operations
  return(value) [optional]
}
```

```
is_triangular <- function(y){
  n <- (sqrt((8 * y) + 1) - 1) / 2
  if (as.integer(n) == n) {
    return(TRUE)
  }
  return(FALSE)
}
```

```
source("triangular.R")
is_triangular(91)
is_triangular(78)
is_triangular(4)
is_triangular(56)
```

```
find_triangular <- function(max_number){
  to_test <- 1:max_number
  triangular_numbers <- numeric(0)
  for (i in to_test) {
    if (is_triangular(i)) {
      triangular_numbers <- c(triangular_numbers, i)
    }
  }
  print(paste("There are", length(triangular_numbers),
              "triangular numbers between 1 and ", max_number))
  return(triangular_numbers)
}
```

```
source("triangular.R")
find_triangular(100)
find_triangular(1000)
find_triangular(10000)
```

```
tell_fortune <- function() {
  if (runif(1) < 0.3) {
    print("Today is going to be a great day for you!")
  } else {
    print("You should have stayed in bed")
  }
}
order_three <- function(x, y, z) {
  return(sort(c(x, y, z)))
}
order_three_list <- function(x, y, z) {
  my_ord <- sort(c(x, y, z))
  return(list("First" = my_ord[1],
              "Second" = my_ord[2],
              "Third" = my_ord[3]))
}
split_string <- function(s, separator = "_") {
  return(strsplit(s, separator)[[1]])
}
```

## 8.13 Importing Libraries

```
library([NAME_OF_PACKAGE])
```

## 8.14 Random Numbers

```
runif(3)
rnorm(4, mean = 1, sd = 5)
rpois(4, lambda = 5)
```

```
v <- 1:10
sample(v, 2)
sample(v, 11, replace = TRUE)
```

## 8.15 Vectorize It!

```
M <- matrix(runif(10000 * 10000), 10000, 10000)
get_row_means <- function(M) {
system.time(get_row_means(M))
system.time(rowMeans(M))
```

```
MList <- as.list(rep(NA, length = 20))
randmat <- function(x) {
    return(matrix(rnorm(25), 5, 5))
}
MList <- lapply(MList, randmat)
Meig <- lapply(MList, function(x)
    return(eigen(x, only.values = TRUE)$values[1]))
print(unlist(Meig))
DNAlist <- list(A = 'GTTTCG',
                B = 'GCCGCA',
                C = 'TTATAG',
                D = 'CGACGA')
count_nucl <- function(seq, nucl) {
    pos <- gregexpr(pattern = nucl, text = seq)[[1]]
    if(pos[1] == "-1") {
        return(0)
    } else {
        return(length(pos))
    }
}
numAs <- sapply(DNAlist, count_nucl, nucl = 'A')
print(numAs)
numGs <- sapply(DNAlist, count_nucl, nucl = 'G')
print(numGs)
```

## 8.16 Debugging

```
myfun <- function(i, x) {
  for (z in 1:i) {
    x <- x * 2
    browser()
  }
  return(x)
}
```

```
myfun(5, 2)
```

## 8.17 Interfacing with the Operating System

```
system("wc -l < ../data/H938_Euro_chr6.geno")
```

```
numlines <- system("wc -l < ../data/H938_Euro_chr6.geno", intern = TRUE)
numlines
```

```
mydf <- read.table(file = textConnection(
    system("grep 'rs125283' ../data/H938_Euro_chr6.geno",
    intern = TRUE)))
mydf
```

## 8.18 Running R from the Command Line

```
Rscript my_script_file.R
```

```
args <- commandArgs(TRUE)
num.args <- length(args)
print(paste("Number of command-line arguments:", num.args))
if (num.args > 0) {
    for (i in 1:num.args) {
        print(paste(i, "->", args[i]))
    }
}
input.file <- "test.txt"
number.replicates <- 10
starting.point <- 3.14
if (num.args >= 1) {
    input.file <- args[1]
}
if (num.args >= 2) {
    number.replicates <- as.integer(args[2])
}
if (num.args >= 3) {
    starting.point <- as.double(args[3])
}
print(c(input.file, number.replicates, starting.point))
```

```
Rscript my_script.R abc.txt 5 100.0
Rscript my_script.R abc.txt 5
Rscript my_script.R abc.txt
Rscript my_script.R
```

## 8.19 Statistics in R

```
attach(iris)
str(iris)
```

```
levels(Species)
```

```
summary(iris)
table(Species)
table(Species, Petal.Width)
table(Species, Petal.Width)
```

```
t.test(Sepal.Width[Species == "setosa"],
```

```
linearmod <- lm(Sepal.Width ~ Sepal.Length)
summary(linearmod)
```

## 8.20 Basic Plotting

### *8.20.1 Scatter Plots*

```
x <- 1:30
y <- rnorm(30, mean = x)
y2 <- rnorm(30, mean = x, sd = sqrt(x))
plot(y ~ x)
plot(x, y)
plot(x, y, type = "l")
plot(x, y, type = "b")
plot(x, y, type = "b", pch = 4)
plot(x, y, type = "b", pch = 4, col = "blue")
abline(c(0, 1))
points(x, y2, col = "red")
plot(x, y2, col = "orange", xlab = "my x-label", ylab =  "yyy")
plot(x, y2, xlim = c(1,10))
```

### *8.20.2 Histograms*

```
d1 <- rpois(100, lambda = 3)
hist(d1)
hist(d1, breaks = 4)
hist(d1, breaks = c(0, 1, 3, 5, 7, 11, 21))
hist(d1, freq = TRUE)
hist(d1, freq = FALSE)
z <- hist(d1, plot = FALSE)
z$counts
z$mids
```

### *8.20.3 Bar Plots*

```
data(islands)
barplot(islands)
barplot(islands, horiz = TRUE)
barplot(islands, horiz = TRUE, las = 1)
data(iris)
barplot(height = iris$Petal.Width, beside = TRUE, col = iris$Species)
```

### *8.20.4 Box Plots*

```
data(iris)
boxplot(iris$Petal.Width ~ iris$Species, col = c("red", "green", "blue"))
```

### *8.20.5 3D Plotting (in 2D)*

```
x <- sort(rnorm(100))
y <- sort(rnorm(50))
z <- x %o% y
image(z)
filled.contour(z)
```

## 8.21 Finding Packages for Biological Research

## 8.22 Documenting Code

````
```{r}
Everything between these marks will be
evaluated in the R console
```
Text not enclosed in triple tick marks is not evaluated.
This is the spot to write extensive documentation.
You can name your code chunks:
```{r histogram}
hist(iris$Sepal.Width)
```
````

## 8.23 Exercises

### *8.23.1 Self-Incompatibility in Plants*

### *8.23.2 Body Mass of Mammals*

```
system.time(source("body_mass_family.R"))
```

### *8.23.3 Leaf Area Using Image Processing*

### *8.23.4 Titles and Citations*

## 8.24 References and Reading

### *Books*

### *Online Resources*

CHAPTER 9

# Data Wrangling and Visualization

9.1 Efficient Data Analysis and Visualization

9.2 Welcome to the **tidyverse**

```
install.packages("tidyverse")
```

```
library(tidyverse)
```

```
tidyverse_update()
```

```
tidyverse_packages()
```

### *9.2.1 Reading Data*

```
library(tidyverse)
setwd("CSB/data_wrangling/sandbox/")
popsize <- read_tsv("../data/FauchaldEtAl2017/pop_size.csv")
```

### *9.2.2 Tibbles*

```
popsize
```

```
head(popsize, 3)
tail(popsize, 3)
```

```
View(popsize)
```

```
glimpse(popsize)
```

```
dim(popsize)
nrow(popsize)
ncol(popsize)
```

## 9.3 Selecting and Manipulating Data

```
ndvi <- read_tsv("../data/FauchaldEtAl2017/ndvi.csv")
head(ndvi)
```

### *9.3.1 Subsetting Data*

```
select(ndvi, Herd, NDVI_May)
```

```
select(ndvi, Herd:NDVI_May)
select(ndvi, -Herd, -Year)
select(ndvi, -(Year:NDVI_May))
select(ndvi, matches("NDV"))
```

```
filter(popsize, Herd == "WAH")
filter(popsize, Year >= 1970, Year <= 1980)
filter(popsize, Year %in% c(1970, 1980, 1990))
```

```
slice(popsize, 20:30)
top_n(popsize, 10, Pop_Size)
top_n(popsize, 15, desc(Pop_Size))
sample_n(popsize, 5)
sample_frac(popsize, 0.02)
```

### *9.3.2 Pipelines*

```
unique(popsize[order(popsize$Herd), 1])
```

```
select(popsize, Herd)
distinct(select(popsize, Herd))
arrange(distinct(select(popsize, Herd)), Herd)
```

```
popsize %>% select(Herd) %>% distinct() %>% arrange (Herd)
```

```
herds <- popsize %>%
```

### *9.3.3 Renaming Columns*

```
popsize %>% rename(h = Herd)
```

```
rename(popsize, h = Herd)
```

### *9.3.4 Adding Variables*

```
ndvi %>% mutate(meanNDVI = (NDVI_May + NDVI_June_August) / 2) %>% head(4)
```

```
ndvi %>% transmute(meanNDVI = (NDVI_May + NDVI_June_August) / 2) %>% head
    (4)
```

## 9.4 Counting and Computing Statistics

### *9.4.1 Summarize Data*

```
ndvi %>% summarise(mean_May = mean(NDVI_May))
```

```
ndvi %>% summarise(mean_May = mean(NDVI_May),
    sd_May = sd(NDVI_May),
    median_May = median(NDVI_May))
```

### *9.4.2 Grouping Data*

```
popsize %>% group_by(Herd) %>%
    summarise(avgPS = mean(Pop_Size),
    minPS = min(Pop_Size),
    maxPS = max(Pop_Size)) %>%
    arrange(Herd)
```

```
popsize %>% group_by(Herd) %>% tally() %>% arrange(Herd)
```

```
popsize %>%
    group_by(Herd) %>%
    summarise(tot = n()) %>%
    arrange(Herd)
```

```
popsize %>%
    group_by(Herd) %>%
    mutate(zscore = (Pop_Size - mean(Pop_Size)) / sd(Pop_Size))
```

```
popsize %>%
    group_by(Herd) %>%
    mutate(zscore = scale(Pop_Size))
```

## 9.5 Data Wrangling

### *9.5.1 Gathering*

```
seaice <- read_tsv("../data/FauchaldEtAl2017/sea_ice.csv")
head(seaice)
```

```
seaice %>% gather(Month, Cover, 3:14)
```

```
seaice <- seaice %>% gather(Month, Cover, 3:14)
```

### *9.5.2 Spreading*

```
head(popsize, 3)
```

```
popsize %>%
    filter(Year > 1979, Year < 1985) %>%
    spread(Year, Pop_Size)
```

```
popsize %>%
    filter(Year > 1979, Year < 1985) %>%
    spread(Year, Pop_Size, fill = "")
```

#### *9.5.3 Joining Tibbles*

```
combined <- inner_join(popsize, ndvi)
head(combined, 4)
dim(popsize)
dim(ndvi)
dim(combined)
```

```
cor(combined %>% select(Pop_Size, NDVI_May))
```

## 9.6 Data Visualization

#### *9.6.1 Philosophy of `ggplot2`*

#### *9.6.2 The Structure of a Plot*

```
library(tidyverse)
popsize <- read_tsv("../data/FauchaldEtAl2017/pop_size.csv")
ndvi <- read_tsv("../data/FauchaldEtAl2017/ndvi.csv")
seaice <- read_tsv("../data/FauchaldEtAl2017/sea_ice.csv")
snow <- read_tsv("../data/FauchaldEtAl2017/snow.csv")
seaice <- seaice %>% gather(Month, Cover, 3:14)
```

```
ggplot(data = popsize)
```

```
ggplot(data = popsize) + aes(x = Year, y = Pop_Size, colour = Herd)
```

```
ggplot(data = popsize) +
    aes(x = Year, y = Pop_Size, colour = Herd) +
    geom_point() +
    geom_line()
```

### *9.6.3 Plotting Frequency Distribution of One Continuous Variable*

```
ggplot(data = ndvi) + aes(x = NDVI_May) + geom_histogram()
```

```
ggplot(data = ndvi) + aes(x = NDVI_May) + geom_histogram()
ggplot(data = ndvi, aes(x = NDVI_May)) + geom_histogram()
ggplot(data = ndvi) + geom_histogram(aes(x = NDVI_May))
```

```
ggplot(data = ndvi) + aes(x = NDVI_May) + geom_density()
```

### *9.6.4 Box Plots and Violin Plots*

```
pl <- ggplot(data = ndvi) + aes(x = Herd, y = NDVI_May)
pl + geom_boxplot()
pl + geom_violin()
```

```
pl + geom_boxplot() + aes(fill = Herd)
```

### *9.6.5 Bar Plots*

```
ggplot(data = seaice %>% filter(Herd == "WAH")) +
    aes(x = Year) +
    geom_bar()
```

```
ggplot(data = seaice %>% filter(Herd == "WAH", Year == 1990)) +
    aes(x = Month, y = Cover) +
    geom_col()
```

```
seaice$Month <- factor(seaice$Month, month.abb)
ggplot(data = seaice %>%
    filter(Herd == "WAH", Year == 1990)) +
    aes(x = Month, y = Cover) +
    geom_col()
```

### *9.6.6 Scatter Plots*

```
pl <- ggplot(data = popsize %>%
    filter(Herd == "WAH")) +
    aes(x = Year, y = Pop_Size) +
    geom_point()
show(pl)
```

```
pl + geom_smooth()
```

```
pl + geom_smooth(method = "lm")
pl + geom_smooth(method = "lm", formula = y ~ poly(x, 3), se = FALSE)
```

### *9.6.7 Plotting Experimental Errors*

```
stats <- popsize %>% filter(Herd %in% c("GRH", "PCH")) %>%
          group_by(Herd) %>%
          summarise(
                meanPopSize= mean(Pop_Size),
                SD = sd(Pop_Size),
                N = n(),
                SEM = SD/sqrt(N),
                CI = SEM * qt(0.975, N-1))
ggplot(data = stats) +
        aes(x = Herd, y = meanPopSize) +
        geom_col()
```

```
 limits <- aes(ymax = stats$meanPopSize + stats$CI,
               ymin = stats$meanPopSize - stats$CI)
# plot including confidence intervals
ggplot(data = stats) +
        aes(x = Herd, y = meanPopSize) +
        geom_col() +
        geom_errorbar(limits, width = .3)
```

### *9.6.8 Scales*

```
pl <- ggplot(data = popsize,
        aes(x = Herd, y = Pop_Size, fill = Herd)) +
        geom_boxplot()
show(pl)
```

```
pl + scale_fill_brewer(palette = "Set3")
pl + scale_fill_hue()
pl + scale_fill_manual(values = rainbow(11), name = "aaa")
```

```
pl <- ggplot(data = seaice %>% filter(Herd == "BEV")) +
        aes(x = Year, y = Month, colour = Cover,
        size = Cover) +
        geom_point()
show(pl)
```

```
pl + scale_colour_gradient(high = "white", low = "red")
```

### *9.6.9 Faceting*

```
ggplot(data = seaice %>%
                filter(Herd %in% c("WAH", "BAT"),
                Year %in% c(1980, 1990, 2000, 2010))) +
        aes(x = Month, y = Cover) +
        geom_col() +
        facet_grid(Year~Herd)
```

```
ggplot(data = seaice %>% filter(Year == 2010)) +
        aes(x = Month, y = Cover) +
        geom_col() +
        facet_wrap(~Herd)
```

```
ggplot(data = seaice %>% filter(Year == 2010)) +
         aes(x = Month, y = Cover) +
         geom_col() +
         facet_wrap(~Herd, scales = "free")
```

### *9.6.10 Labels*

```
pl <- ggplot(data = popsize) +
        aes(x = Year, y = Pop_Size) +
        geom_point()
pl + xlab("Year")
pl + ylab("Population Size")
pl + ggtitle("Population Dynamics")
```

### *9.6.11 Legends*

```
pl <- ggplot(data = popsize) +
        aes(x = Herd, y = Pop_Size, fill = Herd) +
        geom_boxplot()
show(pl)
pl + theme(legend.position = "bottom")
pl + theme(legend.position = "top")
pl + theme(legend.position = "none")
```

```
pl <- ggplot(data = popsize) +
    aes(x = Year, y = Pop_Size, colour = Herd,
    alpha = sqrt(Pop_Size)) +
    geom_point()
show(pl)
pl + guides(colour = guide_legend(nrow = 4,
pl + guides(colour = "none")
```

### *9.6.12 Themes*

```
snow <- read_tsv("../data/FauchaldEtAl2017/snow.csv")
pl <- ggplot(data = snow %>%
        filter(Herd == "CAH"),
        aes(y = Week_snowmelt, x = Perc_snowcover)) +
        geom_point()
show(pl)
pl + theme_bw()
pl + theme_linedraw()
pl + theme_minimal()
```

```
library(ggthemes)
show(pl + theme_wsj())
show(pl + theme_fivethirtyeight())
```

### *9.6.13 Setting a Feature*

```
pl <- ggplot(data = popsize) +
        aes(x = Year, y = Pop_Size, colour = Herd) +
        geom_point()
pl <- ggplot(data = popsize) +
        aes(x = Year, y = Pop_Size) +
        geom_point(colour = "red")
```

### *9.6.14 Saving*

```
ggsave(filename = "../sandbox/test.pdf", plot = pl, width = 3, height = 4)
```

## 9.7 Tips & Tricks

```
popsize %>%
        filter(Year > 1979, Year < 1985) %>%
        spread(Year, Pop_Size) %>%
        select(Herd, `1980`)
```

```
popsize %>%
        group_by(Herd, Year) %>%
        tally() %>%
        ungroup() %>%
        summarise(n = sum(n))
```

```
ndvi %>% mutate(maxndvi = max(NDVI_May, NDVI_June_August)) %>% head(4)
```

```
ndvi %>%
        rowwise() %>%
        mutate(maxndvi = max(NDVI_May,
        NDVI_June_August)) %>% head(4)
```

## 9.8 Exercises

### *9.8.1 Life History in Songbirds*

### *9.8.2 Drosophilidae Wings*

### *9.8.3 Extinction Risk Meta-Analysis*

## 9.9 References and Reading

### *Books*

### *Online Resources*

CHAPTER 10

# Relational Databases



```
sqlite3
```

## 10.5 Getting Started with SQLite

### *10.5.1 Comments*

### *10.5.2 Data Types*

### *10.5.3 Creating and Importing Tables*

```
.mode csv
.import ../data/Lohr2015_data.csv lohr
.save lohr.db
.exit
```

```
sqlite3 lohr.db
```

### *10.5.4 Basic Queries*

```
.tables
```

```
.schema
```

```
SELECT * FROM lohr LIMIT 4;
```

```
.mode column
.header on
.width 5 5 5 4 4 4 4
SELECT * FROM lohr LIMIT 4;
```

```
SELECT size, pop FROM lohr LIMIT 5;
```

```
SELECT DISTINCT size FROM lohr;
```

```
SELECT DISTINCT pop FROM lohr;
```

```
SELECT DISTINCT pop FROM lohr ORDER BY pop ASC;
```

```
SELECT DISTINCT ro FROM lohr ORDER BY ro LIMIT 3 OFFSET 10;
```

```
SELECT DISTINCT ro FROM lohr ORDER BY CAST(ro AS INTEGER) DESC LIMIT 3;
```

```
SELECT clone, pop, size FROM lohr WHERE size = "Large" LIMIT 5;
```

```
SELECT ro FROM lohr WHERE CAST(ro AS INTEGER) > 140 AND CAST(ro AS INTEGER)
   < 142;
```

```
SELECT CAST(ro AS INTEGER) AS ronum FROM lohrWHERE ronum > 140 AND ronum <
   142;
```

```
SELECT DISTINCT pop FROM lohr WHERE pop LIKE "%T";
SELECT DISTINCT pop FROM lohr WHERE pop LIKE "B%";
SELECT DISTINCT pop FROM lohr WHERE pop LIKE "A___";
```

```
SELECT DISTINCT pop FROM lohr WHERE pop GLOB "K*";
SELECT DISTINCT pop FROM lohr WHERE pop GLOB "?1?";
```

```
SELECT size, AVG(CAST(ad AS INTEGER)) AS avglifespan FROM lohr GROUP BY
   size;
```

```
SELECT pop, AVG(CAST(ad AS INTEGER)) AS avglifespan FROM lohr GROUP BY pop;
```

```
SELECT pop, COUNT(pop) AS nind FROM lohr GROUP BY pop;
```

```
SELECT pop, COUNT(pop) AS nind FROM lohr GROUP BY pop HAVING nind > 200
   ORDER BY nind DESC;
```

## 10.6 Designing Databases

## 10.7 Working with Databases

### *10.7.1 Joining Tables*

```
.open ../data/Comte2016.db
.schema
```

```
.mode line
SELECT * FROM trans INNER JOIN site ON trans.SiteID = site.SiteID LIMIT 1;
```

```
.mode column
SELECT anguilla, COUNT(anguilla) AS num FROM trans WHERE Per1 = "P1" AND
   Per2 = "P2" GROUP BY anguilla;
```

```
SELECT Basin, anguilla, COUNT(anguilla) FROM site INNER JOIN trans ON site.
   SiteID = trans.SiteID WHERE Per1 = "P1" AND Per2 = "P2" GROUP BY Basin,
   anguilla;
```

### *10.7.2 Views*

```
CREATE VIEW both AS SELECT * FROM site INNER JOIN trans ON site.SiteID =
   trans.SiteID;
```

```
.tables
```

### *10.7.3 Backing Up and Restoring a Database*

```
.output [PATHTOFILE]/sqlitedumpfile.sql
.dump
.exit
```

```
head sqlitedumpfile.sql
```

```
sqlite3 my_database.db < sqlitedumpfile.sql
sqlite3 my_database.db
```

```
.read sqlitedumpfile.sql
```

### *10.7.4 Inserting, Updating, and Deleting Records*

```
INSERT INTO mytable SELECT col1, col2, ..., colN FROM othertable;
```

```
UPDATE trans SET anguilla = "persisted" WHERE anguilla = "Persisted";
```

```
DELETE FROM mytable WHERE mycondition;
```

```
DROP TABLE mytable;
```

```
DROP VIEW myview;
```

#### *10.7.5 Exporting Tables and Views*

```
.mode list
.separator ,
.output test.csv
SELECT DISTINCT Basin, SiteID FROM site;
.output
```

## 10.8 Scripting

```
sqlite3 my_script.sql
```

## 10.9 Graphical User Interfaces (GUIs)

## 10.10 Accessing Databases Programmatically

#### *10.10.1 In Python*

```
#!/usr/bin/python
import sqlite3
con = sqlite3.connect('../data/Comte2016.db')
cursor = con.execute("SELECT DISTINCT Basin FROM site")
for row in cursor:
   print("Basin ", row[0])
con.close()
```

### *10.10.2 In R*

```
install.packages("RSQLite")
library(RSQLite)
sqlite <- dbDriver("SQLite")
con <- dbConnect(sqlite, "../data/Comte2016.db")
results <- dbGetQuery(con, "SELECT DISTINCT Basin FROM site")
print(results)
dbDisconnect(con)
```

## 10.11 Exercises

### *10.11.1 Species Richness of Birds in Wetlands*

### *10.11.2 Gut Microbiome of Termites*

## 10.12 References and Reading