Benchmarking Moments++

Gustavo V. Barroso

2023-06-07

Running moments++

Running moments.LD

```
import moments.LD
import demes
import sys, os
import numpy as np

def simulate_ld(f, sampled_demes):
    g = demes.load(f)
    u = g.metadata["mutation"]["rate"]
    r = g.metadata["recombination"]["rate"]

    y = moments.LD.LDstats.from_demes(g, sampled_demes, r=r, u=u)
    return y

def get_parameters(g, Ne=1e4):
    u = g.metadata["mutation"]["rate"]
    r = g.metadata["recombination"]["rate"]
    theta = 4 * Ne * u
    rho = 4 * Ne * r
    pop_ids = []
    nus = []
    for d in g.demes:
        pop_ids.append(d.name)
        nus.append(d.epochs[0].start_size / Ne)
    num_pops = len(pop_ids)
    m = np.zeros((num_pops, num_pops))
    for mig in g.migrations:
        source_idx = pop_ids.index(mig.source)
        dest_idx = pop_ids.index(mig.dest)
        rate = mig.rate
        m[dest_idx, source_idx] = 2 * Ne * rate
    return pop_ids, nus, theta, rho, m

def write_column(y, out_file):
    mld, mh = moments.LD.Util.moment_names(y.num_pops)
    # write D2
```

```
    for m, v in zip(mld, y[0]):
        if m.split("_")[0] == "DD":
            out_file.write(str(v) + "\n")
    # write Dz
    for m, v in zip(mld, y[0]):
        if m.split("_")[0] == "Dz":
            out_file.write(str(v) + "\n")
    # write H
    for m, v in zip(mh, y[-1]):
        out_file.write(str(v) + "\n")
    # write pi2
    for m, v in zip(mld, y[0]):
        if m.split("_")[0] == "pi2":
            out_file.write(str(v) + "\n")

for i in range(1, 13):
    with open(f"one_epoch/model_{i}_steady_state_LD.txt", "w+") as out_file:
        yaml_file = f"one_epoch/model_{i}.yaml"
        g = demes.load(yaml_file)
        pop_ids, nus, theta, rho, m = get_parameters(g)
        y = moments.LD.LDstats.steady_state(nus, m=m, rho=rho, theta=theta, pop_ids=pop_ids)
        write_column(y, out_file)

for i in range(1, 13):
  with open(f"multi_epoch/moments_LD_model_{i}.csv", "w+") as fout:
      f = f"multi_epoch/model_{i}.yaml"
      pops = ["A", "B", "C"]
      y = simulate_ld(f, pops)
      stats = moments.LD.Util.moment_names(len(pops))
      l = ",".join(stats[0] + stats[1]) + "\n"
      fout.write(l)
      l = ",".join([str(_) for _ in y[0]] + [str(_) for _ in y[1]]) + "\n"
      fout.write(l)
```

```
for i in range(13, 15):
  with open(f"multi_epoch/moments_LD_model_{i}.csv", "w+") as fout:
      f = f"multi_epoch/model_{i}.yaml"
      pops = ["A", "B", "C", "D"]
      y = simulate_ld(f, pops)
      stats = moments.LD.Util.moment_names(len(pops))
      l = ",".join(stats[0] + stats[1]) + "\n"
      fout.write(l)
      l = ",".join([str(_) for _ in y[0]] + [str(_) for _ in y[1]]) + "\n"
      fout.write(l)
```

```
with open(f"multi_epoch/moments_LD_model_15.csv", "w+") as fout:
    f = f"multi_epoch/model_15.yaml"
    pops = ["A", "B", "C", "D", "E"]
    y = simulate_ld(f, pops)
    stats = moments.LD.Util.moment_names(len(pops))
    l = ",".join(stats[0] + stats[1]) + "\n"
```

```
    fout.write(l)
    l = ",".join([str(_) for _ in y[0]] + [str(_) for _ in y[1]]) + "\n"
    fout.write(l)
```

Plots

```
# 3 sampled populations
stats <- read.table("multi_epoch/model_1_expectations.txt") %>% select(V1)
stats <- stats[-31,] # deletes Dummy moment

num_models <- 12
mpp <- as.data.frame(matrix(nrow=52, ncol=num_models))
for(i in 1:num_models) {
  mpp[,i] <- read.table(paste("multi_epoch/model_", i, "_expectations.txt", sep="")) %>% select(V3)
}

names(mpp) <- paste("m++", 1:num_models, sep="")
mpp <- slice(mpp, 1:24, 32:52, 25:30) # to match moments.LD output

py <- as.data.frame(matrix(nrow=num_models, ncol=51))
for(i in 1:num_models) {
  py[i,] <- read.csv(paste("multi_epoch/moments_LD_model_", i, ".csv", sep=""))
}

py <- as_tibble(t(py))
names(py) <- paste("py_", 1:num_models, sep="")
py <- type_convert(py)

ratios <- select(mpp, starts_with("m")) / select(py, starts_with("py"))
names(ratios) <- paste("m", 1:num_models, sep="")
ratios$stat <- stats

molten_ratios <- pivot_longer(ratios, cols=starts_with("m"))

p1 <- ggplot(data=molten_ratios, aes(x=stat, y=value, shape=name))
p1 <- p1 + geom_point(size=3) + theme_bw()
p1 <- p1 + geom_hline(yintercept=0.99, linetype=2)
p1 <- p1 + geom_hline(yintercept=1.01, linetype=2)
p1 <- p1 + labs(title="++ / LD", x="Moment idx", y="Ratio", shape="Model")
p1 <- p1 + scale_shape_manual(values=(0:11))
p1 <- p1 + theme(axis.title=element_text(size=12),
                 axis.text=element_text(size=10),
                 axis.text.x=element_text(angle=90, size=8, vjust=0.5, hjust=1.0),
                 legend.position="bottom")

save_plot("multi_epoch/ratios_3-pops.pdf", p1, base_height=12, base_width=12)

# 4 sampled populations (models m5-m6)
mpp_m13 <- read.table("multi_epoch/model_13_expectations.txt") %>% select(!V2)
mpp_m14 <- read.table("multi_epoch/model_14_expectations.txt") %>% select(V3)
```

```
momspp <- bind_cols(mpp_m13, mpp_m14)
names(momspp) <- c("stat", "m13++", "m14++")
momspp <- slice(momspp, 1:50, 62:116, 51:60) # re-orders rows to match moments.LD

py_m13 <- read.csv("multi_epoch/moments_LD_model_13.csv")
py_m14 <- read.csv("multi_epoch/moments_LD_model_14.csv")

py_moms <- as_tibble(cbind(names(py_m13), t(bind_rows(py_m13, py_m14))))
names(py_moms) <- c("stat", "m13py", "m14py")
py_moms <- type_convert(py_moms)

ratios <- select(momspp, starts_with("m")) / select(py_moms, starts_with("m"))
ratios <- bind_cols(momspp$stat, ratios)
names(ratios) <- c("stat", "m13", "m14")

molten_ratios <- pivot_longer(ratios, cols=starts_with("m"))
p2 <- ggplot(data=molten_ratios, aes(x=stat, y=value, shape=name))
p2 <- p2 + geom_point(size=3) + theme_bw()
p2 <- p2 + geom_hline(yintercept=0.99, linetype=2)
p2 <- p2 + geom_hline(yintercept=1.01, linetype=2)
p2 <- p2 + labs(title="++ / LD", x="Moment", y="Ratio", shape="Model")
p2 <- p2 + scale_shape_manual(values=c(0, 1))
p2 <- p2 + theme(axis.title=element_text(size=12),
                 axis.text=element_text(size=10),
                 axis.text.x=element_text(angle=90, size=8, vjust=0.5, hjust=1.0),
                 legend.position="bottom")

save_plot("multi_epoch/ratios_4-pops.pdf", p2, base_height=15, base_width=15)

# 5 sampled populations (model m7)
momspp <- read.table("multi_epoch/model_15_expectations.txt") %>% select(!V2)
names(momspp) <- c("stat", "m15++")
momspp <- slice(momspp, 1:90, 107:226, 91:105) # to match moments.LD

py_moms <- as_tibble(t(read.csv("multi_epoch/moments_LD_model_15.csv")))
names(py_moms) <- "m15py"

ratios <- select(momspp, starts_with("m")) / select(py_moms, starts_with("m"))
ratios <- bind_cols(momspp$stat, ratios)
names(ratios) <- c("stat", "m15")

molten_ratios <- pivot_longer(ratios, cols=starts_with("m"))
p3 <- ggplot(data=molten_ratios, aes(x=stat, y=value))
p3 <- p3 + geom_point(size=3) + theme_bw()
p3 <- p3 + geom_hline(yintercept=0.99, linetype=2)
p3 <- p3 + geom_hline(yintercept=1.01, linetype=2)
p3 <- p3 + labs(title="++ / LD", x="Moment", y="Ratio")
p3 <- p3 + theme(axis.title=element_text(size=12),
                 axis.text=element_text(size=10),
                 axis.text.x=element_text(angle=90, size=8, vjust=0.5, hjust=1.0),
                 legend.position="bottom")

save_plot("multi_epoch/ratios_5-pops.pdf", p3, base_height=18, base_width=18)
```

```
# these are 3-pop models
stats <- read.table("one_epoch/model_1_expectations.txt") %>% select(V1)
stats <- stats[-31,] # deletes Dummy moment
num_models <- 12

# steady state
mpp <- as.data.frame(matrix(nrow=52, ncol=num_models))
for(i in 1:num_models) {
  mpp[,i] <- read.table(paste("one_epoch/model_", i, "_steady_state.txt", sep=""))
}
names(mpp) <- paste("m", 1:num_models, sep="")
mpp <- mpp[-31,] # deletes Dummy moment

py <- as.data.frame(matrix(nrow=num_models, ncol=51))
for(i in 1:num_models) {
  py[i,] <- t(read.csv(paste("one_epoch/model_", i, "_steady_state_LD.txt", sep=""), header=F))
}

py <- as_tibble(t(py))
names(py) <- paste("py_", 1:num_models, sep="")
py <- type_convert(py)

ratios <- select(mpp, starts_with("m")) / select(py, starts_with("py"))
names(ratios) <- paste("m", 1:num_models, sep="")
ratios$stat <- stats

molten_ratios <- pivot_longer(ratios, cols=starts_with("m"))
molten_ratios$r <- rep(c(rep(1e-4, 3), rep(1e-5, 3), rep(1e-6, 3), rep(1e-7, 3)), 51)
molten_ratios$m_AB <- rep(rep(c(1e-3, 1e-4, 1e-5), 4), 51)

p4 <- ggplot(data=molten_ratios, aes(x=stat, y=value, shape=name))
p4 <- p4 + geom_point(size=3) + theme_bw() + facet_grid(r~m_AB)
p4 <- p4 + geom_hline(yintercept=0.99, linetype=2)
p4 <- p4 + geom_hline(yintercept=1.01, linetype=2)
p4 <- p4 + labs(title="++ / LD for r (rows) x m_AB (cols)", x="Moment idx", y="Ratio", shape="Model")
p4 <- p4 + scale_shape_manual(values=(0:11))
p4 <- p4 + theme(axis.title=element_text(size=12),
                 axis.text=element_text(size=10),
                 axis.text.x=element_text(angle=90, size=8, vjust=0.5, hjust=1.0),
                 legend.position="bottom")

save_plot("one_epoch/steady_state_3-pops.pdf", p4, base_height=18, base_width=18)
```

Check-my-matrix

```
import moments.LD
import demes
import csv
import numpy as np

np.set_printoptions(threshold=10000)
```

```
def pulse_migration(num_pops, pop0, pop1, f):
    """
    Pulse migration from pop0 into pop1 with proportion f.
    """
    # create the matrix that would build a new population via admixture
    A = moments.LD.Matrices.admix_ld(num_pops, pop0, pop1, f)
    # the last population created replaces pop1
    mom_from = moments.LD.Util.moment_names(num_pops + 1)[0]
    mom_to = moments.LD.Util.moment_names(num_pops)[0]
    P = np.zeros((len(mom_to), len(mom_from)))
    for i, m in enumerate(mom_to):
        l = m.split("_")
        for k in range(1, len(l)):
            if l[k] == str(pop1):
                l[k] = str(num_pops)
        m_from = "_".join(l)
        m_from = moments.LD.Util.map_moment(m_from)
        j = mom_from.index(m_from)
        P[i, j] = 1
    return P.dot(A)

# some hard coding never hurt anyone...
mat_LD = pulse_migration(3, 0, 1, 0.3)
matpp = np.loadtxt('multi_epoch/model_2_e_6_admix.csv', delimiter=",")

print(((matpp - mat_LD)).sum())
print(((matpp - mat_LD)**2).sum())

mat_LD = pulse_migration(4, 0, 3, 0.2)
matpp = np.loadtxt('multi_epoch/model_3_e_5_admix.csv', delimiter=",")
print(((matpp - mat_LD)).sum())
print(((matpp - mat_LD)**2).sum())

for i in range(len(mat_LD)):
    if np.abs(mat_LD[i] - matpp[i]).sum() > 1e-15:
        print(i, moments.LD.Util.moment_names(4)[0][i], np.abs(mat_LD[i] - matpp[i]).sum())
        print(mat_LD[i])
        print(matpp[i])

mat_LD = pulse_migration(5, 2, 4, 0.5)
matpp = np.loadtxt('multi_epoch/model_15_e_9_admix.csv', delimiter=",")
print(((matpp - mat_LD)).sum())
print(((matpp - mat_LD)**2).sum())

for i in range(len(mat_LD)):
    if np.abs(mat_LD[i] - matpp[i]).sum() > 1e-15:
        print(i, moments.LD.Util.moment_names(5)[0][i], np.abs(mat_LD[i] - matpp[i]).sum())
        print(mat_LD[i])
        print(matpp[i])

mig_mat = [
        [0, 1e-4, 0],
        [0, 0, 0],
```

```
        [0, 0, 0]
        ]

matpp = np.loadtxt('multi_epoch/model_2_e_5_mig.csv', delimiter=",")
mat_LD = moments.LD.Matrices.migration_ld(3, 2 * mig_mat).todense()
#print(matpp - mat_LD)
print(((matpp - mat_LD)).sum())
print(((matpp - mat_LD)**2).sum())

for i in range(len(mat_LD)):
    if np.abs(mat_LD[i] - matpp[i]).sum() > 1e-15:
        print(i, moments.LD.Util.moment_names(5)[0][i], np.abs(mat_LD[i] - matpp[i]).sum())
        print(mat_LD[i])
        print(matpp[i])

mig_mat = [
        [0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0],
        [0, 0, 1e-4, 0, 0]
        ]

matpp = np.loadtxt('multi_epoch/model_15_e_10_mig.csv', delimiter=",")
mat_LD = moments.LD.Matrices.migration_ld(5, 2 * mig_mat).todense()
#print(matpp - mat_LD)
print(((matpp - mat_LD)).sum())
print(((matpp - mat_LD)**2).sum())

for i in range(len(mat_LD)):
    if np.abs(mat_LD[i] - matpp[i]).sum() > 1e-15:
        print(i, moments.LD.Util.moment_names(5)[0][i], np.abs(mat_LD[i] - matpp[i]).sum())
        print(mat_LD[i])
        print(matpp[i])
```