Appendix 2: Brownian Motion Simulation

Chris Bentz

December 11, 2020

Load Packages

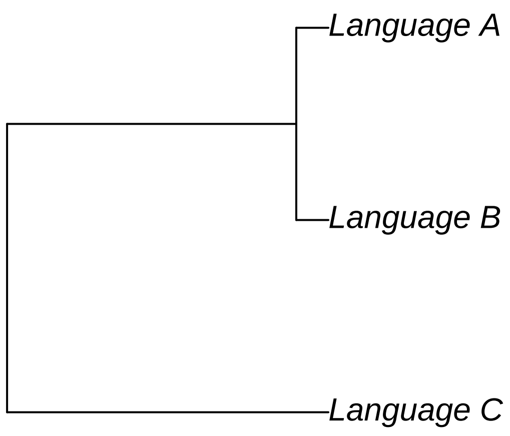
Load packages. If they are not installed yet on your local machine, use install.packages() to install them.

```r
library(ape)
library(phytools)
library(GGally)
library(tidyr)
library(plyr)
library(scales)
library(rstatix)
```

Phylogenetic Tree

Provide a phylogenetic tree. We here build a simple tree with three taxa "manually" by supplying a text string in Newick format.

```r
# create Newick tree as text string
text.string <- "(Language_C:10, (Language_B:1, Language_A:1):9);"
tree <- read.tree(text = text.string) # conversion to object of type ``list''
plot(tree)
```



Simulate Brownian Motion along the Phylogenetic Tree

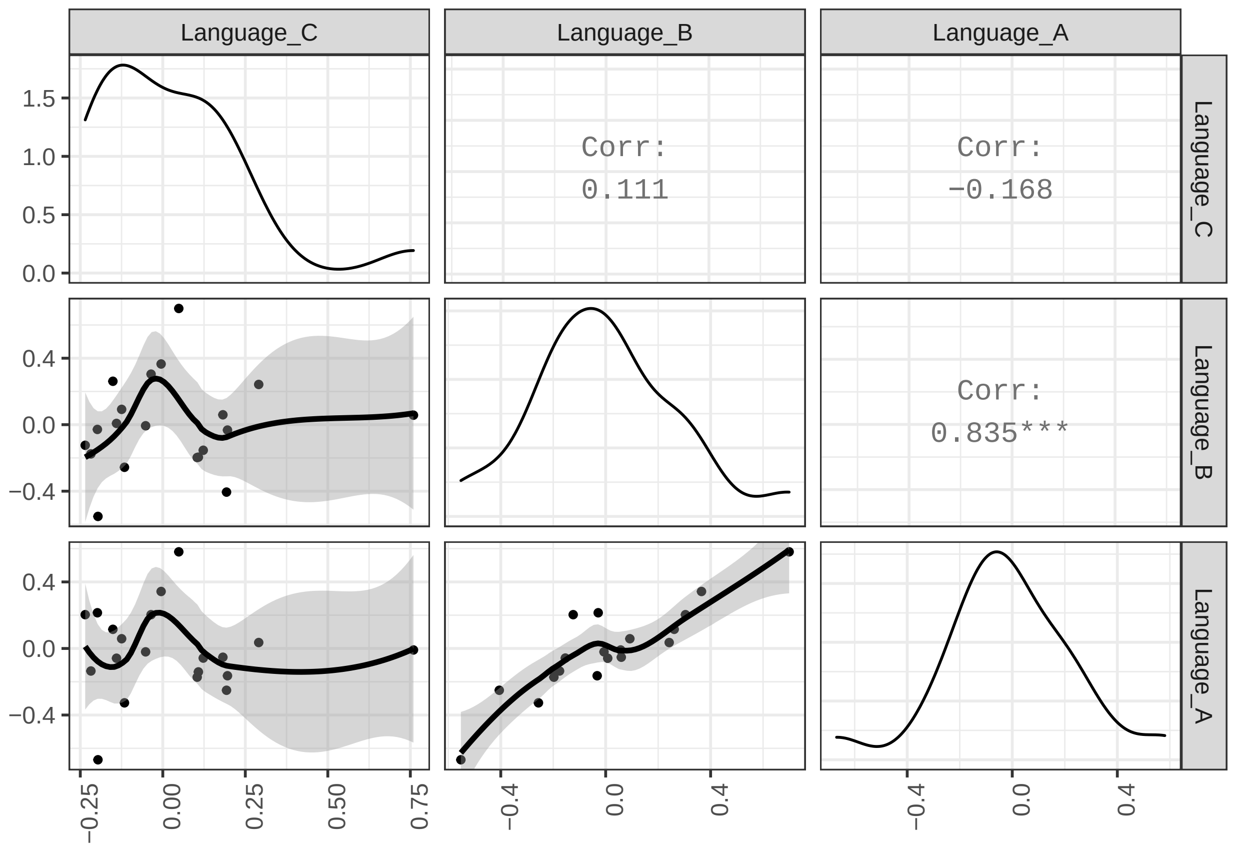
We here simulate the evolution of trait values (i.e. pseudo-complexity values in our case) via the process of Brownian motion along the branches of our phylogenetic tree. The number of simulations per taxon (i.e. language) is set as "nsim". This is conceptualized as the number of pseudo-complexity measurements per language. For further explanations of the function fastBM() see Revell (2016).

```
# define variance of diffusion process
sig2 = 0.01 # this variance setting is copied from Revell (2016)
# define number of BM simulations
nsim = 20
# set the seed for random number generation in order to get the same result
# when the code is re-run
set.seed(1)
# simulate Brownian motion on a tree with fastBM()
simulation <- fastBM(tree, mu = 0, sig2 = sig2, internal = F, nsim = nsim)
```

Correlation Plots

It is expected that the trait values for taxa which form a clade (i.e. language A and B) are positively correlated due to the shared history. While for trait values of taxa without shared history (i.e. language A and C and language B and C) there is no correlation expected. This can be checked with a correlation plot. Importantly, however, this does not mean that language A and B are expected to have average trait values which are different from the average trait value of C.

```
simulation.df <- as.data.frame(t(simulation))
# the t() function is transposing the rows and columns
cor.plot <- ggpairs(simulation.df,
                        lower = list(continuous = wrap("smooth_loess", size = 1)),
                        upper = list(continuous = wrap('cor', method = "spearman"))) +
                        theme_bw() +
                        theme(axis.text.x = element_text(angle = 90, hjust = 1))
print(cor.plot)
```



Safe figure to file.

```
ggsave("Figures/BM_CorPlot.pdf", cor.plot, dpi = 300, scale = 1,
       device = cairo_pdf)
```

```
## Saving 5 x 5 in image
```

Visualization: Density Distributions

Plot density distributions of BM generated pseudo-complexity measurements by language. Individual values for each complexity pseudo-measurement are plotted as black dots.

Transform data frame from wide format to long format (this is necessary for later plotting and analyses by columns rather than rows).

```
simulation.df.long <- gather(simulation.df, key = language, value = value, Language_A:Language_C)
head(simulation.df.long)
```

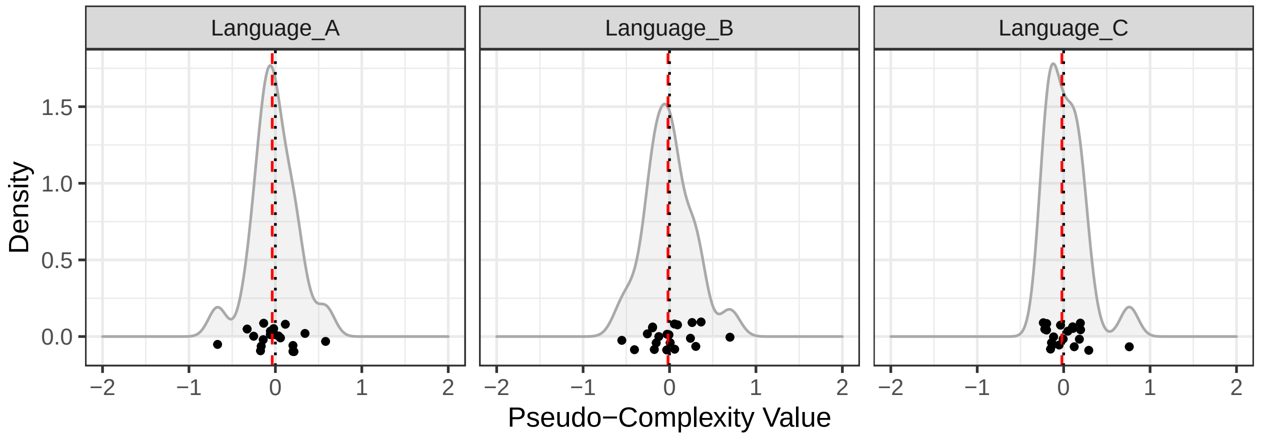
```
##     language       value
## 1 Language_A  0.21462108
## 2 Language_A -0.17230804
## 3 Language_A -0.05263219
## 4 Language_A -0.66890333
## 5 Language_A  0.34254100
## 6 Language_A  0.03570572
```

Get mean, median, and standard deviation values.

```
# get mean values for each language
mu <- ddply(simulation.df.long, "language", summarise, grp.mean = mean(value, na.rm = T))
# get median values for each language
med <- ddply(simulation.df.long, "language", summarise, grp.median = median(value, na.rm = T))
# get standard deviation values for each language
sdev <- ddply(simulation.df.long, "language", summarise, grp.sd = sd(value, na.rm = T))
```

Plot density distributions with indication of median (mean) values.

```
density.plot <- ggplot(simulation.df.long, aes(x = value)) +
 # histograms could be added as well here (or instead of the density distributions)
 # geom_histogram(aes(y = ..density..), colour = "white", fill = "light grey",
                # binwidth = 0.1) +
 geom_density(alpha = .2, fill = "grey", color = "darkgrey") +
 geom_jitter(data = simulation.df.long, aes(x = value, y = 0),
             size = 1, height = 0.1, width = 0) + # add some jitter to prevent overplotting
 facet_wrap(~ language) +
 # geom_vline(data = mu, aes(xintercept=grp.mean),
 #           linetype = "dotted", color = "blue") +
 geom_vline(data = med, aes(xintercept = grp.median),
             linetype = "dashed", color = "red") +
 geom_vline(aes(xintercept = 0), linetype = "dotted") +
 labs(x = "Pseudo-Complexity Value", y = "Density") +
 xlim(-2, 2) +
 theme_bw()
print(density.plot)
```



Safe figure to file.

```
ggsave("Figures/BM_simulation_densities.pdf", density.plot, dpi = 300, scale = 1,
       device = cairo_pdf)
```

```
## Saving 7 x 2.5 in image
```

Descriptive Statistics

Give an overview of mean, median, and standard deviation values (i.e. values reflecting the location of a distribution).

```r
stats.df <- cbind(mu, med[, 2], sdev[, 2])
colnames(stats.df) <- c("language", "mu", "med", "sdev")
stats.df.sorted <- stats.df[order(-stats.df$med), ]
# round values to two decimal places,  the "-1" excludes column 1
stats.df.sorted[, -1] <- round(stats.df.sorted[, -1], 2)
print(stats.df.sorted)
```

```
##     language    mu   med sdev
## 2 Language_B  0.00 -0.02 0.28
## 3 Language_C  0.03 -0.02 0.24
## 1 Language_A -0.02 -0.04 0.26
```

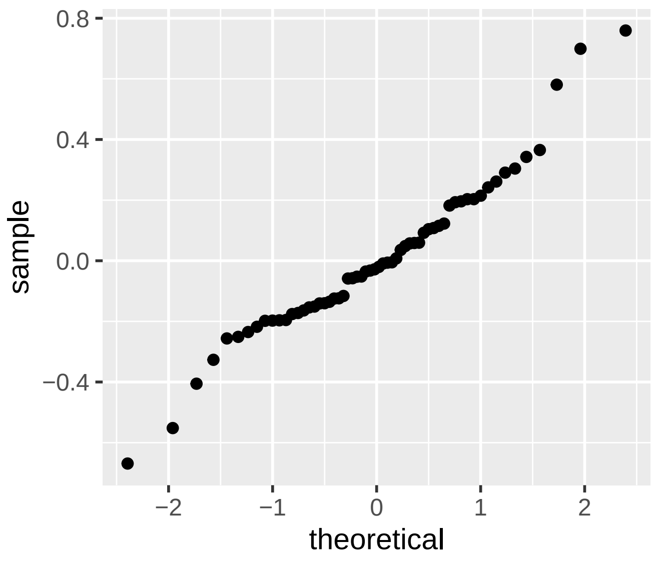
Output data frame as csv file.

```r
write.csv(stats.df.sorted, file = "Tables/BMsimulation_descriptiveStats.csv", row.names = F)
```

Normality

The assumption that the tested data stems from a normally distributed population is often necessary for the mathematical proofs underlying standard statistical techniques. We might apply normality tests to check for this assumption (e.g. Baayen 2008, p. 73), but some statisticians advice against such pre-tests, since they are often too sensitive (MacDonald 2014, p. 133-136, Rasch et al. (2020), p. 67). In fact, Rasch et al. (2020, p. xi) argue based on earlier simulation studies that almost all standard statistical tests are fairly robust against deviations from normality. However, it is still advisable to check for gross deviations from normality in the data. One common way of doing this is quantile-quantile plots. The points should here roughly follow a straight line (Crawley 2007, p. 281).

```r
ggplot(simulation.df.long, aes(sample = value)) + stat_qq()
```



Statistical Tests

Standard t-tests can be used to assess significant differences in the means of the pseudo-complexity distributions, if we assume that the underlying population distributions are normal. Wilcoxon tests are a non-parametric alternative, i.e. they do not make assumptions about the underlying population distribution (Crawley 2007,

p. 283; Baayen 2008, p. 77). We here run pairwise t-tests. If we supply two data vectors, then by default the function pairwise.t.test() runs a Welch two sample t-test (for unpaired samples); with the argument "paired = T" a paired t-test is invoked. We here assume that our data consists of two vectors which are linked via the same measurement procedure(s), and we hence consider them "paired". A more general term is "related samples", which are defined as "two sets of data where a data point in one set has a pairwise relationship to a point in the other set of data" (Cahusac 2021, p. 56).

P-value adjustment for multiple comparisons: In case of multiple testing, we should account for the fact that the likelihood of finding a significant result by chance increases with the number of statistical tests. One of the most conservative methods to account for this is the so-called Bonferroni correction, i.e. multiplying the p-values with the number of tests. This method assumes that tests are independent of one another (MacDonald 2014, p. 254-260). However, since we here compare, for example, language A to B and language A to C, there is dependence between the test results. We therefore apply the so-called Holm-Bonferroni method, which is less conservative. It does not assume independence between tests (see the descriptions in the vignette invoked by the command "?p.adjust()").

```r
p.values <- pairwise.t.test(simulation.df.long$value, simulation.df.long$language,
                  paired = T, p.adjust.method = "holm")
p.values
```

```
##
##  Pairwise comparisons using paired t tests
##
## data:  simulation.df.long$value and simulation.df.long$language
##
##            Language_A Language_B
## Language_B 1          -
## Language_C 1          1
##
## P value adjustment method: holm
```

Effect sizes

Statistical significance is only one part of the story. For instance, a difference in complexity values might be statistically significant, but so small that it is negligible for any theorizing. In fact, it is sometimes argued that effect sizes - rather than p-values - should be the aim of statistical inquiry (Cahusac 2020, p. 12-15). An overview of effect size measures per statistical test is given in Patil (2020). In conjunction with the t-test we here use Cohen's d (i.e. function cohens_d() of the "rstatix" package). (Note: the d estimate given by this function can be negative. However, the sign is not relevant here, only the absolute value.)

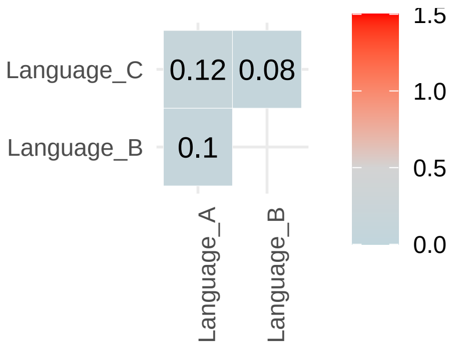
```r
effect.sizes <- cohens_d(simulation.df.long, value ~ language, paired = T)
print(effect.sizes)
```

```
## # A tibble: 3 x 7
##   .y.   group1     group2     effsize    n1    n2 magnitude
## * <chr> <chr>      <chr>        <dbl> <int> <int> <ord>
## 1 value Language_A Language_B -0.0979    20    20 negligible
## 2 value Language_A Language_C -0.118     20    20 negligible
## 3 value Language_B Language_C -0.0825    20    20 negligible
```

Effect size heatmap

Plot a heatmap with effect sizes to get a better overview.

```
effect.sizes.plot <- ggplot(as.data.frame(effect.sizes), aes(group1, group2)) +
  geom_tile(aes(fill = abs(effsize)), color = "white") +
  scale_fill_gradient2(low = "light blue", mid = "light grey", high = "red",
                       midpoint = 0.5, limit = c(0,1.5)) +
  geom_text(aes(label = round(abs(effsize), 2))) +
  labs(x = "", y = "") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
effect.sizes.plot
```



Safe figure to file.

```
ggsave("Figures/BMsimulation_effectSizes.pdf", effect.sizes.plot, dpi = 300, scale = 1,
       device = cairo_pdf)
```

```
## Saving 3 x 2 in image
```

Interpretation

Note that the exact results of the tests above will differ every time the random vectors are re-sampled (which is avoided here by using the set.seed() function). Having said this, we do not seem to find any significant differences between the distributions of values generated along the branches of the tree by Brownian motion. In fact, it follows from the core properties of (the basic) Brownian motion model that the expected value of a trait at any time is equal to the value of the trait at time zero (Harmon 2019, p. 41). This means we should not expect to see significant differences in the average trait values (pseudo-complexities) of the languages in this simple simulation.

References

Baayen, R. H. (2008). Analyzing linguistic data: A practical introduction using statistics in R. Cambridge University Press.

Cahusac, P. M. B. (2021). Evidence-based statistics. John Wiley & Sons.

Crawley, M. J. (2007). The R book. John Wiley & Sons Ltd.

Ehret, K. and B. Szmrecsanyi (2016). An information-theoretic approach to assess linguistic complexity. In: R. Baechler & G. Seiler (eds.), Complexity, Isolation, and Variation, 71-94. Berlin: de Gruyter.

Harmon, L. (2019). Phylogenetic comparative methods. Online at https://lukejharmon.github.io/pcm/ (last accessed 25/11/2020).

Kruschke, J. K. (2012). Bayesian estimation supersedes the t test. Journal of Experimental Psychology.

McDonald, J.H. (2014). Handbook of Biological Statistics (3rd ed.). Sparky House Publishing, Baltimore, Maryland. online at http://www.biostathandbook.com

Patil, I. (2020). Test and effect size details. online at https://cran.r-project.org/web/packages/statsExpressions/vignettes/stats_details.html.

Rasch, D., Verdooren, R., and J. Pilz (2020). Applied statistics. Theory and problem solutions with R. John Wiley & Sons Ltd.

Revell, L. (2016). Simulating Brownian motion in R. Online at http://www.phytools.org/Bariloche2016/ex/3/Simulating-BM.html (last accessed 25/11/2020).