Paleoelevation records from lipid biomarkers: Application to the tropical Andes

V.J. Anderson et al.

- 2015180_SupplementalData.xlsx
Appendix

1. Representative demagnetization plots

Subachoque samples, shown in N-S orthographic view. AF steps 50-800 shown. Tickmark scale on axes is $10^5$ Gauss. On the left is sample 2SUB7C, shown as an example of a well-behaved sample with an MAD = 4.2°; on the right is sample 2SUB19B, shown as an example of a more problematic sample, with MAD = 19.4°. Subachoque samples generally were the most strongly magnetized, and behaved well under AF demagnetization.

Guasca samples, shown in N-S orthographic view. AF steps 50-800 shown. Tickmark scale on axes is $10^6$ Gauss. On the left is sample 2GUA9C, shown as an example of a well-behaved sample with an MAD = 14.6°; on the right is sample 2GUA6C, shown as an example of a more problematic sample, with MAD = 15.6°. Guasca samples tended to be significantly more weakly magnetized than at Subachoque, and as a result the measured directions were a bit more scattered, but still generally demagnetized cleanly with AF.
Tequendama samples, shown in N-S orthographic view. Thermal steps NRM-375 °C shown. Tickmark scale on axes is $10^{-7}$ Gauss. On the left is sample 1TEQT18C, with thermal steps NRM-375 °C shown, and MAD = 9.5°. On the right is sample 1TEQT25D, shown as an example of a more problematic sample, with thermal steps NRM-225 °C shown and MAD = 18.2°. The samples in the Tequendama section were generally characterized by extremely weak magnetization, and low unblocking temperatures, as is shown in these two samples.
2. Method for classifying paleomagnetic samples

At each stratigraphic level, 3 or 4 cores were taken; paleomagnetic measurements were taken on each core. In order to classify the quality of the data at each site, we determined the angular distance between each pair of points using the dot product. Each paleomagnetic direction (declination = \( \phi \), inclination = \( \theta \)) was converted into Cartesian unit vectors using the following formulas:

\[
\begin{align*}
x &= \sin(\phi) \cos(\theta) \\
y &= \cos(\phi) \cos(\theta) \\
z &= \sin(\theta)
\end{align*}
\]

Then, the angle \( \alpha \) between measured paleomagnetic directions was given by

\[
\alpha = \cos^{-1} \left( \frac{v_1 \cdot v_2}{|v_1||v_2|} \right) = \cos^{-1}(v_1 \cdot v_2)
\]

We computed the angular difference between all possible pairs of points, and classified them according to the scheme diagrammed below. "Good" samples were designated in cases where all points lay within 90 degrees of each other, "Fair" samples were designated in cases when the majority of points lay within 90 degrees, and "Poor" samples were designated when there was no consensus among measured paleomagnetic directions. In cases where measurements were disqualified due to having an MAD > 20 degrees, if only 2 samples remained, the highest classification that they could obtain was a "Fair" if they were within 90 degrees of one another. If only one sample remained, it was automatically classified as a "Poor" sample.

The final measured direction for each stratigraphic interval was computed by taking the average declination and inclination across all measurements. The error on these measurements (as shown in the error bars in Figure 4) is calculated as the average of the angular distances between all pairs of points. Below is the Python (v3.1.2) code used to perform this analysis.
import math as m
import itertools

def find_angle(v1, v2):
    # inputs: [dec, inc] in degrees for two vectors
    # Convert vectors to radians
    v1r = [m.radians(x) for x in v1]
    v2r = [m.radians(x) for x in v2]

    # Convert to cartesian coordinates
    x1 = m.sin(v1r[0])*m.cos(v1r[1])
    x2 = m.sin(v2r[0])*m.cos(v2r[1])
    y1 = m.cos(v1r[0])*m.cos(v1r[1])
    y2 = m.cos(v2r[0])*m.cos(v2r[1])
    z1 = m.sin(v1r[1])
    z2 = m.sin(v2r[1])

    # angle = acos(v1 dot v2)
    dot = x1*x2 + y1*y2 + z1*z2
    angle = m.acos(dot)

    return m.degrees(angle)

fin = open('PMag-all.csv')
all_data = []
header = fin.readline().strip().split('	')

# Read in the data. Text file is in tab-delimited format with
# columns (indices listed in parentheses):
# (0) Section Name, (1) Sample #, (2) Strat level (m),
# (3) Declination (degrees), (4) Inclination (degrees), (5) MAD (degrees)

for line in fin:
    temp = line.strip().split('	')
    templine = []
    for i in range(len(temp)):
        try:
            templine.append(float(temp[i]))

        except ValueError:
            templine.append(temp[i])

    all_data.append(templine)

fin.close()

# Initialize temporary storage
vectors = [all_data[0][3:5]]
cur_strat = all_data[0][2]
summary = []

for i in range(1, len(all_data)):
    if all_data[i][-1] < 20.0:
        if all_data[i][2] == cur_strat:
            vectors.append(all_data[i][3:5])
        else:
            # Find all unique pairs of points
            combos = itertools.combinations(range(len(vectors)), 2)
            angles = []
            # Determine the angle between each measurement point
            for c in combos:
                angles.append(find_angle(vectors[c[0]], vectors[c[1]]))

            # Simplify the angular differences: code 1 for difference < 90, 0 for > 90
            class_vec = [int(a < 90.0) for a in angles]
# Classify each sample based on angular differences.

A = all measurements agree (are within 90 degrees of each other
B = majority of measurements agree (or, only 2 points remain and both agree)
C = 50-50 split, or all measurements scattered.

```python
if len(class_vec) > 3:
    if sum(class_vec) == 6:
        sam_class = 'Good'
    elif sum(class_vec) >= 3:
        sam_class = 'Fair'
    else:
        sam_class = 'Poor'
else:
    if sum(class_vec) == 3:
        sam_class = 'Good'
    elif sum(class_vec) > 0:
        sam_class = 'Fair'
    else:
        sam_class = 'Poor'

try:
    avg_dec = sum([v[0] for v in vectors])/float(len(vectors))
    avg_inc = sum([v[1] for v in vectors])/float(len(vectors))
    avg_angle = sum(angles)/float(len(angles))
except ZeroDivisionError:
    avg_dec = vectors[0][0]
    avg_inc = vectors[0][1]
    avg_angle = -1.0

summary.append((all_data[i-1][0], cur_strat, sam_class, \
                avg_dec, avg_inc, avg_angle))
```

# Reset the sample list for the next entry

```python
cur_strat = all_data[i][2]
vectors = [all_data[i][3:5]]
```

fout = open('pmag_summary.txt', 'w')
```
fout.write('Section	Strat Level	Class	Avg Dec	Avg Inc	Avg Angle
')
```

for i in range(len(summary)):
    for j in range(i, len(summary[i])):
        fout.write(('	' + str(summary[i][j])))
    fout.write('
')

fout.close()
3. Representative chromatograms from selected dD samples

Below are selected examples of chromatograms from samples used in this study. These chromatograms are from the GC-FID analyses that were used to screen the samples prior to isotopic analysis on the GC-IRMS. Additional chromatograms may be obtained by contacting the corresponding author.

Best quality. High abundances, good peak separation, strong odd/even preference:

Medium quality. Moderate to low abundances, fairly strong odd/even preference:

Poor quality. Low abundances, strong odd/even preference, but contains closely eluting peaks that were not removed during urea adduction. Typically these samples were combined with
similar-looking samples from adjacent stratigraphic intervals:

Altered sample – discarded from dataset. High abundances of shorter-chain alkanes, low CPI (0.85):
4. **Results of bootstrapping analysis**

In order to estimate the range of possible cooling supported by our dataset, we used a bootstrapping method, randomly resampling the data (with replacement) from each section, calculating the average for each section and recording the difference between sections (10,000x). Below is a histogram of the results, with the differences between sections on the horizontal axis, and the frequency on the vertical axis.