Brandi Pessman - Data Science Portfolio

2025-02-19

Lincoln, Nebraska, USA | bjpessman@gmail.com | GitHub | LinkedIn

About Me

I am a data scientist with a Ph.D. from the University of Nebraska-Lincoln, where I developed a passion for uncovering patterns in complex datasets and translating them into actionable insights. My expertise lies in data analytics, predictive modeling, pipeline production, and geospatial analysis using tools like R, Python, SQL, high-throughput computing, and QGIS.

I excel at designing scalable workflows and predictive models that transform raw data into resolutions that drive decision-making and innovation. By applying advanced analytical techniques to biological systems, I have developed a deep understanding of complex, dynamic data - skills that translate to solving real-world challenges in business, technology, and environmental science.

Below are selected snippets of projects demonstrating experiences that align with the data scientist role at Penlink.

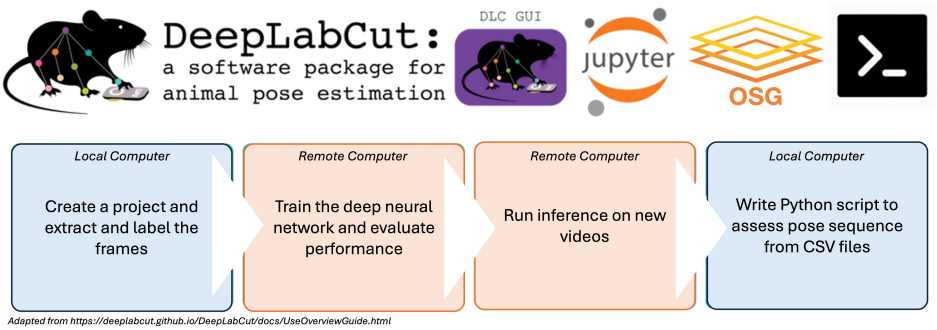
Featured Projects

1. Modeling Leg Movement Through Pattern Sequence Reconstruction

- **Tech Stack**: Python, DeepLabCut, OSPool, HTCondor, High-Throughput Computing
- **Impact**: Applied machine learning and high-throughput computing to track and analyze spider leg movements in mating displays. This work automated behavioral analysis, reducing manual video annotation time and enhancing the study of motion patterns in biomechanics.

– Built and implemented a DeepLabCut, a deep-learning pose estimation software, container to use on the remote servers of the Open Science Pool.

– Utilized machine learning to train the computer to learn key point positions on spider legs during courtship displays.



– Wrote python scripts, shell scripts, and submission files for the HTCondor system.

Python Script

```
import deeplabcut
import os
import sys

video = sys.argv[1]
config = "path"

current_directory = os.getcwd()
config_path = os.path.join(current_directory, config, 'config.yaml')
video_path = os.path.join(current_directory, video)
dest_path = os.path.join(current_directory, config, 'results')

deeplabcut.analyze_videos(config_path, [video_path], save_as_csv=True,
                          destfolder=dest_path)
deeplabcut.filterpredictions(config_path, [video_path], destfolder=dest_path)
```

Shell Script

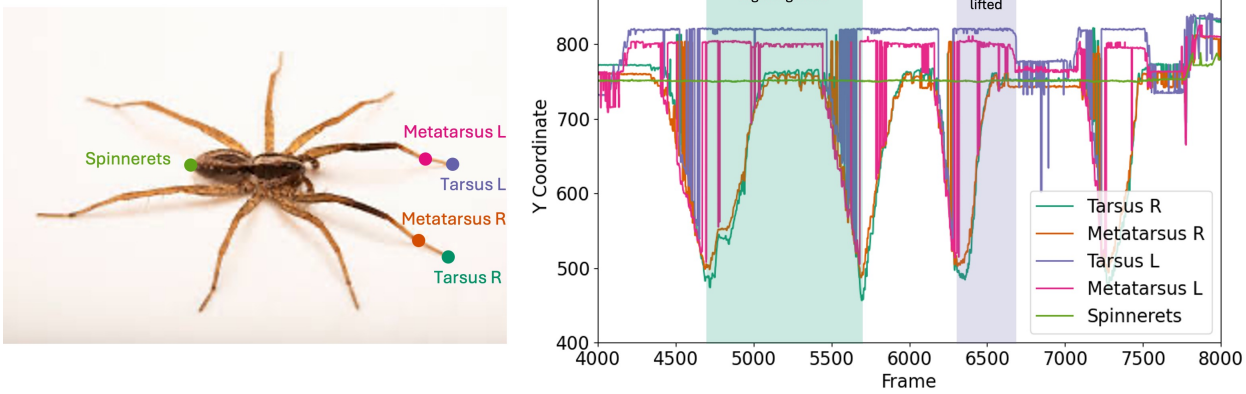
```
video=$1

mkdir -p /tmp/pretrained
ls /usr/local/lib
cp /usr/local/lib/python3.10/dist-packages/deeplabcut/pose_estimation_tensorflow/models/
    pretrained.ORIG/*.yaml /tmp/pretrained
cp /usr/local/lib/python3.10/dist-packages/deeplabcut/pose_estimation_tensorflow/models/
    pretrained.ORIG/*.py /tmp/pretrained

tar -xzf tar3_evaluated.tar.gz -C /srv
mkdir /srv/path/results

if [[ "$video" == *.MP4 ]]; then
    python3.10 deeplabcut_analysis.py $video

else
    echo "Unsupported video format: $video. Please use .MP4."
    exit 1
fi

tar -czf results_${video}.tar.gz path/
```

Submission File

```
video = $Fnx(videopath)
executable = deeplabcut_analysis.sh
arguments = $(video)

transfer_input_files = tar3_evaluated.tar.gz, deeplabcut_analysis.py, osdf://$(videopath)
transfer_output_remaps = "results_$(video).tar.gz = results/results_$(video).tar.gz"

universe = container
```

```
container_image = osdf:///path/DeepLabCut.sif

output = output/output_$(video).out
error = error/error_$(video).err
log = log/log_$(video).log

gpus_minimum_capability = 8.0
request_gpus = 1
request_cpus = 2
request_memory = 40GB
request_disk = 25GB

requirements = regexp("GP-ARGO", GLIDEIN_ResourceName) != True &&
                    (GLIDEIN_ResourceName =!= MY.GLIDEIN_ResourceName0) &&
                    (GLIDEIN_ResourceName =!= MY.GLIDEIN_ResourceName1) &&
                    (GLIDEIN_ResourceName =!= MY.GLIDEIN_ResourceName2)

queue videopath matching path/*.MP4
```

– Processed the resulting CSV file to graph the y-coordinates of different body parts of the spider across all video frames to predict leg movements.



Selected Python Code Behind the Figure

```
import pandas as pd
from pathlib import Path
import numpy as np
import os
import matplotlib.pyplot as plt

Dataframe = pd.read_csv(os.path.join('sg138-19_C0038_filtered.csv'))

df_melted_y = Dataframe.melt(id_vars=['coords'],
                    value_vars=[col for col in Dataframe.columns if col.endswith('.y')],
                    var_name='bodypart',
                    value_name='y')

df_melted_lh = Dataframe.melt(id_vars=['coords'],
                    value_vars=[col for col in Dataframe.columns if col.endswith('.lh')],
```

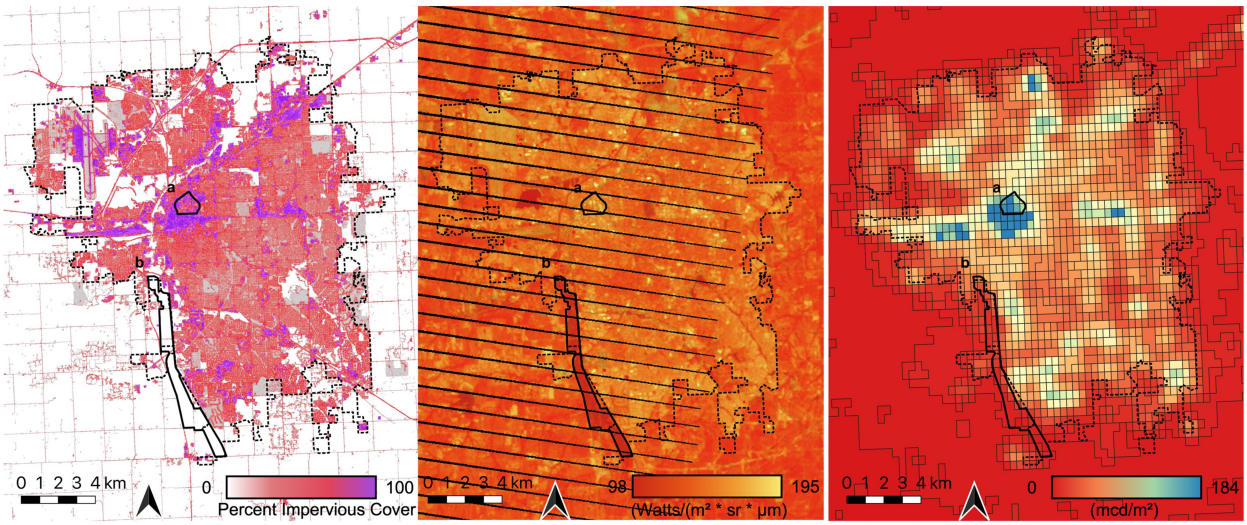
```
                    var_name='bodypart',
                    value_name='lh')
df_melted_lh = df_melted_lh[df_melted_lh['lh'] > 0.5]

merged_df = pd.merge(df_melted_y, df_melted_lh, on=['coords'], how='left')
merged_df = merged_df.dropna()
merged_df = merged_df[merged_df['bodypart_x'].isin(['TR1.y', 'MR1.y', 'TL1.y',
                                                    'ML1.y', 'S.y'])]

plt.figure(figsize=(10, 6))

unique_bodyparts = merged_df['bodypart_x'].unique()
plt.rcParams.update({'font.size': 16})

colors = plt.cm.Dark2(range(len(unique_bodyparts)))

for i, bodypart in enumerate(unique_bodyparts):
    bodypart_data = merged_df[merged_df['bodypart_x'] == bodypart]

    plt.plot(bodypart_data['coords'], bodypart_data['y'], label=bodypart, color=colors[i])

plt.xlabel('Frame')
plt.ylabel('Y Coordinate')

handles, labels = plt.gca().get_legend_handles_labels()
new_labels = ['Tarsus R', 'Metatarsus R', 'Tarsus L', 'Metatarsus L', 'Spinnerets']
plt.legend(handles, new_labels, loc="lower right")

plt.xlim(5000, 8000)
plt.ylim(400, 900)

plt.show()
```

2. Predicting Population Dynamics Through Geospatial Analysis

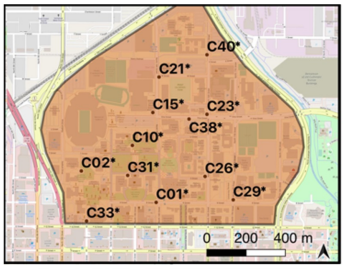
- **Tech Stack**: R, QGIS, Geospatial Analysis, Predictive Modeling
- **Impact**: Used satellite imagery and environmental datasets to predict how environmental conditions impact spider populations. Insights from this study, published in a peer-reviewed journal, inform conservation strategies and demonstrate the power of spatial data in ecological research.
- **Github**: https://github.com/brandipessman/Agelenopsis_aggregation
- **Publication**: https://doi.org/10.1007/s11252-023-01379-z

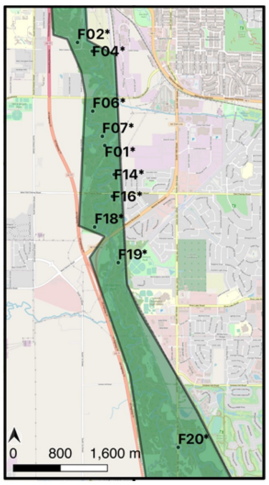
– Used publicly available GIS layers to assess environmental conditions at each of 22 sites (12 in city center - a, and 10 in city forest - b):

- National Land Cover Database - Impervious (building and pavement) Cover 2019
- Landsat-7 World Reference System-2 - Spectral Radiance 2020
- Visible Infrared Imaging Radiometer Suite (VIIRS) - Artificial Night Sky Radiance 2020



– Leveraged QGIS to calculate the average of each variable in a 100m-radius buffer around each site to use in further analyses.





Site	Impervious Cover (%)	Thermal Conditions (Watts/(m² * sr * µm))	Artifical Light at Night (mcd/m²)
C01	68.180447	154.6233	148.890423
C02	62.861889	149.2673	152.636900
C10	60.289828	162.8636	139.000000
C15	82.956538	144.3690	105.836201
C21	82.731251	154.0802	76.976724
C23	85.891998	151.4558	75.494799
C26	80.079451	154.6209	108.000000
C29	80.040861	155.8992	105.568974
C31	66.322875	151.4411	144.420249
C33	85.873050	154.4642	174.574531
C38	79.575990	154.5022	96.359852
C40	83.410740	153.0484	74.999925
F01	0.000000	136.8215	4.347885
F02	1.894909	138.8957	12.293750
F04	0.000000	138.9385	10.000010
F06	0.000000	136.8028	4.559619
F07	0.000000	136.1243	4.400859
F14	5.327918	140.0694	5.000000
F16	0.000000	136.4091	5.742758
F18	0.000000	136.9323	4.490257
F19	0.955806	137.3508	6.020400
F20	4.546412	136.0859	2.000510

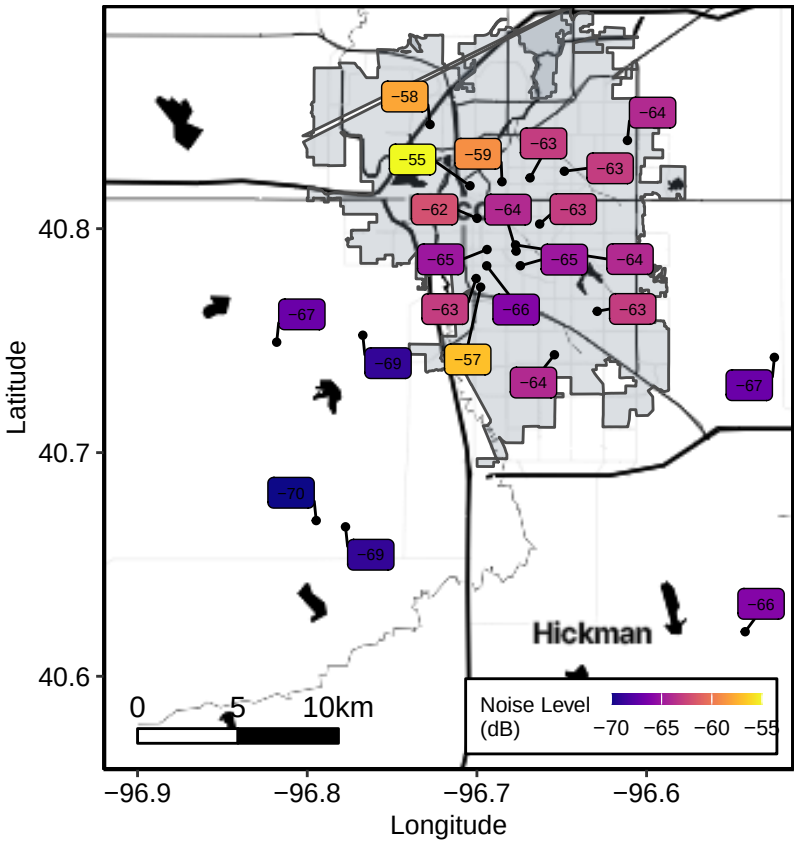
3. Mapping Vibratory Noise Levels in Cities

- **Tech Stack**: R, QGIS, Raven Pro, Principal Component Analysis, Linear Mixed-Effect Models
- **Impact**: Developed spatial models to quantify vibratory noise levels in cities, integrating GIS mapping and statistical modeling. Findings, published in a peer-reviewed journal, provide critical insights into urban noise pollution and its ecological effects.
- **Github**: https://github.com/brandipessman/Vibratory_Noise

- **Publication**: https://doi.org/10.1111/eea.13487

– Recorded vibratory noise levels at 23 sites across urban (city) and rural (countryside) areas in Lincoln, Nebraska and mapped the daily average noise level for each site.

– Utilized the *ggmap* package in R to map the coordinates of the recording sites, integrating data about average noise levels at each site.

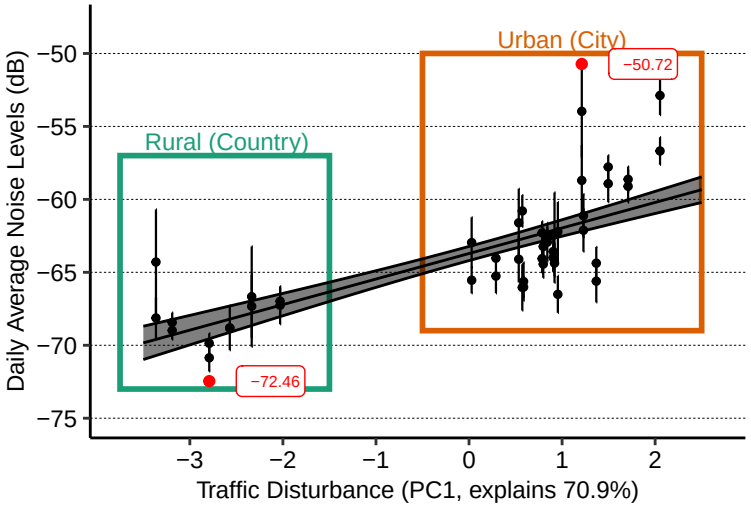


Selected R Code Behind the Map

```
# get background map
lancaster <- map_data("county") %>%
  subset(region == "nebraska") %>%
  subset(subregion == "lancaster")
lc_borders <- c(bottom  = min(lancaster$lat)  + 0.05,
                top     = max(lancaster$lat)  - 0.13,
                left    = min(lancaster$long) - 0.01,
                right   = max(lancaster$long) - 0.05)
map <- get_stadiamap(lc_borders, zoom = 10, maptype = "stamen_toner")

ggmap(map) +
  # add scalebar
  ggsn::scalebar(x.min = -96.9, x.max = -96.5,
                 y.min = 40.57, y.max = 40.899,
                 dist = 5, dist_unit = "km", transform = TRUE,
```

```
                location = "bottomleft",
                st.bottom = FALSE, box.color = c("black", "black"),
                st.size = 4, border.size = 0.5, st.dist = 0.03) +
# add Lincoln outline/shading
geom_polygon(data = lincoln,
             aes(x = longitude, y = latitude),
             fill = "slategray", alpha = 0.25, color = "gray30") +
# add sites and labels
geom_point(data = sites,
           mapping = aes(x = Longitude, y = Latitude),
           pch=21, size = 1, fill = "black") +
# aesthetics
geom_label_repel(aes(x =Longitude, y = Latitude,
                     fill = mean_leq, label = mean_leq),
                     min.segment.length = 0.1, max.overlaps = Inf,
                     size = 2, data = sites) +
scale_fill_viridis(name = "Noise Level \n(dB)", option = "C") +
ylab("Latitude") + xlab("Longitude") +
labs(fill = "Avg. Noise Level (dB)") +
theme_classic() +
theme(panel.border = element_rect(colour = "black", fill=NA, size=1),
      legend.position = c(0.75, 0.07),
      legend.direction = "horizontal",
      legend.background = element_rect(fill = "white",
                                size = 0.5, linetype = "solid",
                                colour ="black"),
      legend.key.height = unit(0.15, 'cm'),
      legend.key.width = unit(0.4, 'cm'),
      text = element_text(size = 10, color = "black", family = "sans"),
      axis.text = element_text(size = 10, color = "black", family = "sans"),
      legend.text = element_text(size = 8, family = "sans"),
      legend.title = element_text(size = 8, family = "sans"))
```

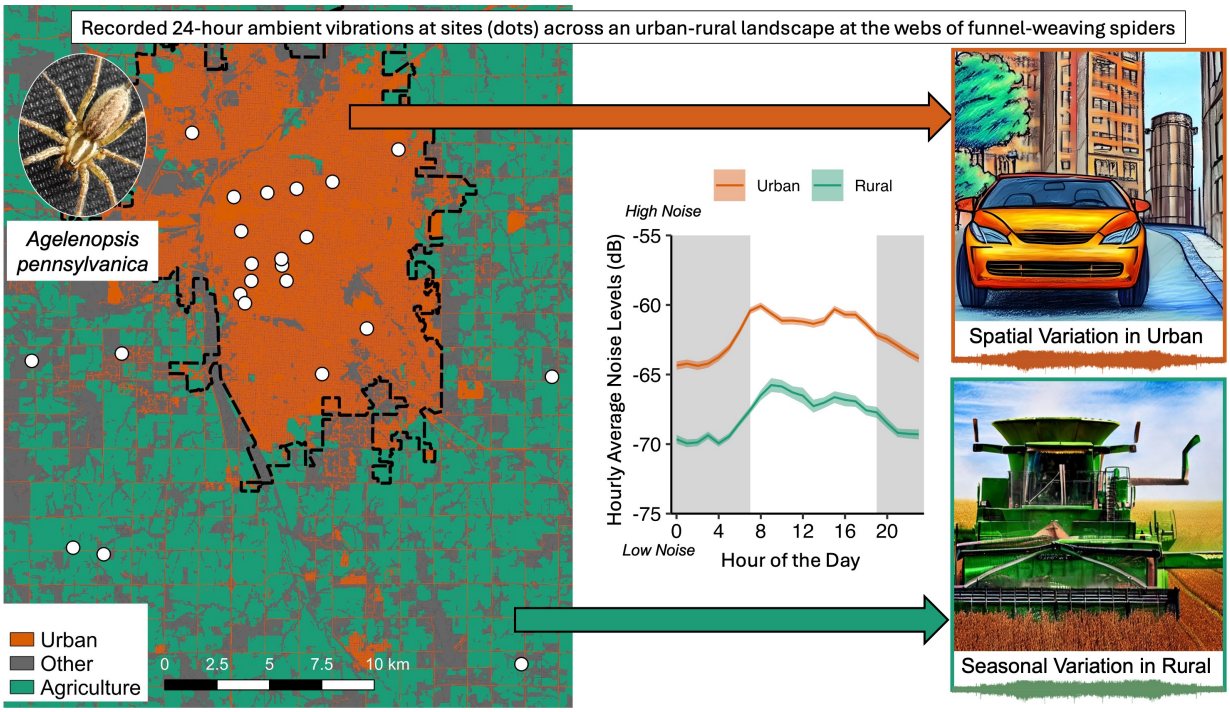
– Performed a Principal Component Analysis (PCA) in R to reduce dimensions of traffic disturbance variables and used a linear mixed-effect model (LMM) to test whether vibratory noise levels correlated with our traffic disturbance principal component.



Selected R Code Building the Model and Calculating the Predictions

```r
noise.lmer <- lmer(noise_levels ~ PC1 + (1|site), data = daily_levels)

predictions <- expand.grid(PC1 = seq(-3.5, 2.5, 0.05))
predictions$response <- predict(noise.lmer, newdata = predictions,
                                se.fit = TRUE, re.form = NA, type = "response")

BootFunc <- function(mm) {
    predict(mm, newdata = predictions, re.form = ~0, type = "response")
}

bigBoot_spatial <- bootMer(noise.lmer, BootFunc, nsim = 1000)
predSE <- t(apply(bigBoot_spatial$t, MARGIN = 2, FUN = sd))
predictions$SE <- predSE[1, ]
```

– Created a graphical summary of the findings that is featured in the publication in a peer-reviewed journal. The map was created using QGIS.



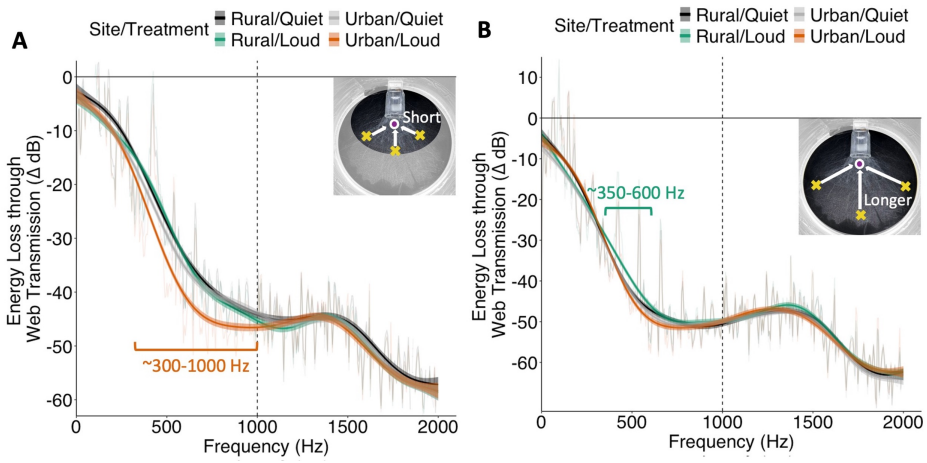
4. Testing Noise Impacts On Vibrations Transmission Through Spider Webs

- **Tech Stack**: R, Raven Pro, Generalized Additive Models, Pairwise Comparisons
- **Impact**: Uncovered how environmental noise influences vibratory properties of spider webs, revealing that spiders may adapt their web-building strategies to optimize signal transmission in noisy conditions.
- **GitHub**: https://github.com/brandipessman/Vibration_Transmission

– Engineered a novel method to measure energy loss as vibrations transmit across spider webs when vibrations were introduced at two distances.

– Tested Generalized Additive Models and pairwise comparisons of the estimated marginal means to compare how webs differed within the range of noise (below 1000 Hz) and outside the range of noise (above 1000 Hz).

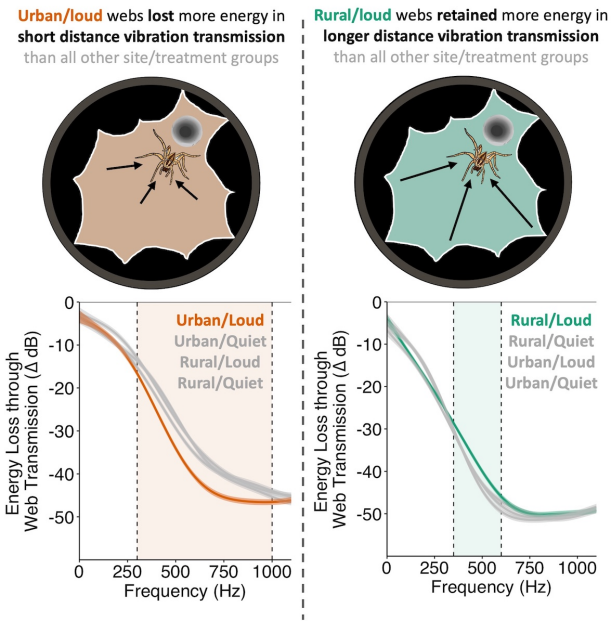
– Employed a fully crossed design where spiders collected from rural (country) and urban (city) constructed webs under quiet (reflecting rural) or loud (reflecting urban) conditions.



R Code Constructing the Model and Calculating Predictions

```
gam_model <- gam(energy_loss ~  Site/Treatment + s(Frequency, by = Site/Treatment, k = 10),
                 data = short)
gam_model_null <- gam(energy_loss ~ 1, data = short)
anova(gam_model_null, gam_model, test = "F")
c(AIC(gam_model), AIC(gam_model_null))
pairs(emmeans(gam_model, ~Site/Treatment))

nd <- data.frame(expand.grid(Frequency = seq(0, 2000, 5),
                             Site = levels(factor(short$Site)),
                             Treatment = levels(factor(short$Treatment))))
nd <- nd %>%
  unite("Site/Treatment", Site, Treatment, sep = "_", remove = FALSE)
predictions <- data.frame(predict(gam_model, newdata = nd, type = "response",
                          se.fit = TRUE))
predictions <- cbind(nd, predictions)
```

– Translated the figure to make a more engaging graphical summary to draw readers to the paper.



5. Assessing the Social Network of High School Students

- **Tech Stack**: SQL
- **Impact**: Built SQL queries to analyze relational data on student friendships, uncovering patterns in social interactions.

– Wrote queries as part of the Stanford Online Course: *Relational Databases and SQL*

Here are the database structures:

Highschooler (ID, name, grade)

There is a high school student with unique ID and a given first name in a certain grade.

Friend (ID1, ID2)

The student with ID1 is friends with the student with ID2. Friendship is mutual, so if (123, 456) is in the Friend table, so is (456, 123).

Likes (ID1, ID2)

The student with ID1 likes the student with ID2. Liking someone is not necessarily mutual, so if (123, 456) is in the Likes table, there is no guarantee that (456, 123) is also present.

Selected exercises

Ex1. For every student who likes someone 2 or more grades younger than themselves, return that student's name and grade, and the name and grade of the student they like.

```
select H1.name as name1, H1.grade as grade1, H2.name as name2, H2.grade as grade2
from Highschooler H1
    join Likes L on H1.ID = L.ID1
    join Highschooler H2 on L.ID2 = H2.ID
where H1.grade - H2.grade > 1
```

Ex2. For every pair of students who both like each other, return the name and grade of both students. Include each pair only once, with the two names in alphabetical order.

name1	grade1	name2	grade2
John	12	Haley	10

```
select H1.name as name1, H1.grade as grade1, H2.name as name2, H2.grade as grade2
from Likes L1
  join Likes L2 on L1.ID1 = L2.ID2 and L1.ID2 = L2.ID1
  join Highschooler H1 on L1.ID1 = H1.ID
  join Highschooler H2 on L1.ID2 = H2.ID
where H1.name < H2.name
```

name1	grade1	name2	grade2
Cassandra	9	Gabriel	9
Jessica	11	Kyle	12

Ex3. Find all students who do not appear in the Likes table (as a student who likes or is liked) and return their names and grades. Sort by grade, then by name within each grade.

```
select name, grade
from Highschooler
where ID not in (select ID1 from Likes) and ID not in (select ID2 from Likes)
order by grade, name
```

name	grade
Jordan	9
Tiffany	9
Logan	12