Cool Quiet City

Federico Cortese

2023-12-12

Competition

This competition's goal is to use contextual data to predict noise distraction and thermal preference across a diversity of indoor and outdoor spaces.

This competition includes the city-scale collection of 9,808 smartwatch-driven micro-survey responses that were collected alongside 2,659,764 physiological and environmental measurements from 98 people using the open-source Cozie-Apple platform combined with geolocation-driven urban digital twin metrics.

Data

The data is available here https://www.kaggle.com/competitions/cool-quiet-city-competition/data

The dataset includes micro-survey responses from 98 people, who each provided at least 100 micro-survey responses from October 2022 to August 2023.

The data collection process for this competition utilized the open-source Cozie Apple platform. This mobile and smartwatch application for iOS devices allows people to complete a watch-based micro-survey and provide real-time feedback about environmental conditions via their Apple Watch. It leverages the built-in sensors of the smartwatch to collect physiological (e.g., heart rate, activity), environmental (sound level), and location (latitude and longitude) data.

The goal of the competition is to predict the values of the columns q_noise_nearby, q_noise_kind, q_earphones, q_thermal_preferenceas well as possible.

Accuracy

The evaluation metric for this competition is accuracy. The metric is calculated as

$$\frac{TP + TN}{TP + TN + FP + FN}$$

As we have four targets here (q_noise_nearby, q_noise_kind, q_earphones, q_thermal_preference), accuracy for each target column is calculated and the average of the four of them is returned.

Prediction

For each id_unique in the test set, you must predict the target variables: q_noise_nearby, q_noise_kind, q_earphones, q_thermal_preference.

Dataset description

The physiological and environmental data were collected at a higher frequency than micro-survey responses. Therefore, these datasets are sparse, i.e., many values are empty as not every row contains a microsurvey response.

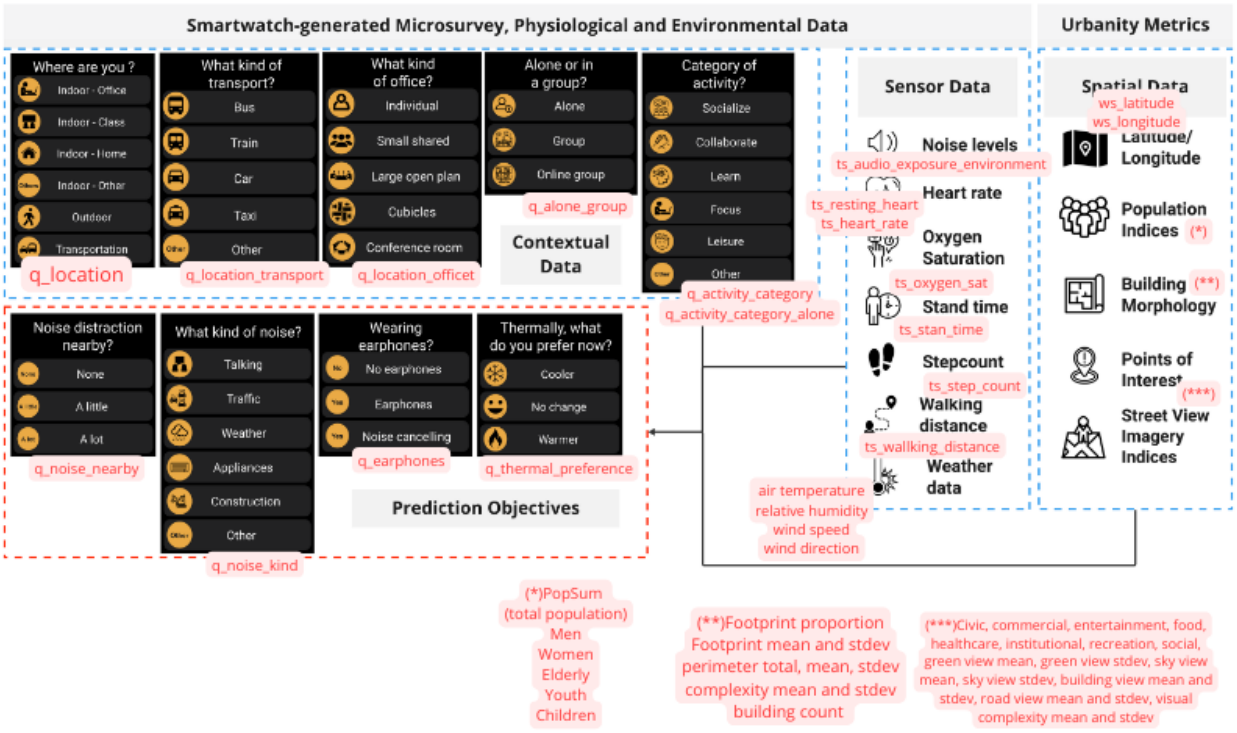


Figure 1: The red small captions refer to the feature names.

The spatial indicators (calculated using the URBANITY PYTHON PACKAGE starting from latitude and longitude) are grouped into Building Morphology and Point of Interest. For more information on the spatial indicators, please refer to Table 1 of Yap, W., Biljecki, F. A Global Feature-Rich Network Dataset of Cities and Dashboard for Comprehensive Urban Analyses. Sci Data 10, 667 (2023).

Urbanity is a network-based Python package to automate the construction of feature rich (contextual and semantic) urban networks at any geographical scale. Through an accessible and simple to use interface, users can request heterogeneous urban information such as street view imagery, building morphology, population (including sub-group), and points of interest for target areas of interest.

Workflow

For our purposes, we need to construct a dataset containing a first column (some initial columns) with observation for the target feature(s) and, subsequently, all features we believe relevant for prediction (we call these explanatory features).

Regarding the cozie dataset, the target features are sparse, so we need to drop all NAs. Then, we need to parse the remaining observations with the "temporally closest" value of summary statistics of the explanatory variables computed based on most "recent" values.

Considering the varying levels of sparsity across datasets, we follow these steps:

1) Conduct spatio-temporal interpolation of weather variables to populate each spatial cell for each time-step.
2) Synthesize this data by computing averages to align with the micro-survey datasets' dimensions. (Hint: we might average environmental observations (through EMAs) between two micro-survey observations).
3) Temporally match these new features with target variables, result: a complete dataset.
4) Train a machine learning algorithm and subsequently conduct accuracy tests.

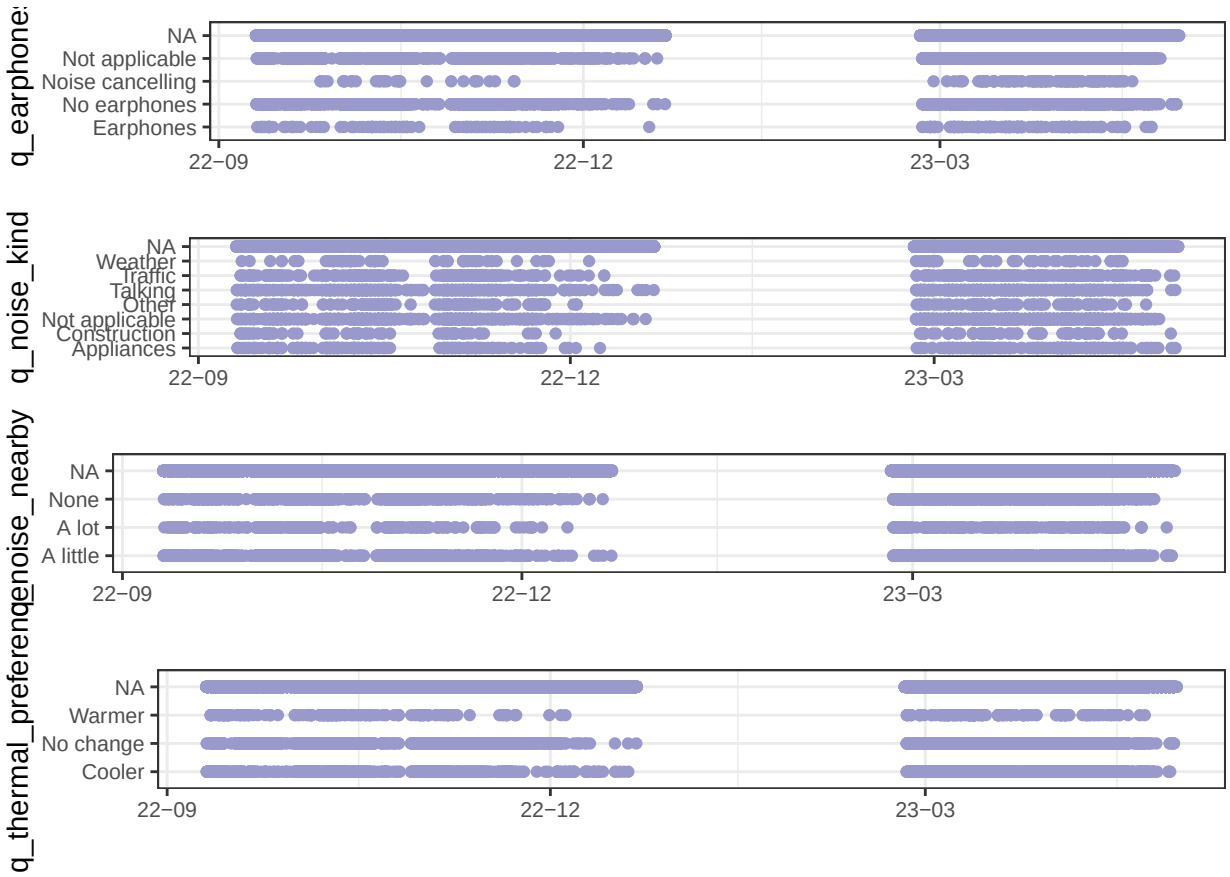
Packages loading

```
# Load packages
library(scales)
library(dplyr)
library(tidyr)
library(tidyverse)
library(ggplot2)
library(ggpubr)
library(lubridate)

load("cleaned_data.Rdata")
```

Target features

```
target_vars=c("q_earphones","q_noise_kind","q_noise_nearby","q_thermal_preference")
target_features_long=cozie_train[,c("time",target_vars)] %>%
  pivot_longer(!time, names_to = "target", values_to = "feedback")

P1=ggplot(data=cozie_train[,c("time",target_vars)], aes(x=time, y = q_earphones)) +
  geom_point(color="#9999CC",na.rm = F) +
  scale_x_datetime(labels = date_format("%y-%m"))+
  theme_bw()+
  labs(x=" ")+
  theme(axis.text=element_text(size=8))

P2=ggplot(data=cozie_train[,c("time",target_vars)], aes(x=time, y = q_noise_kind)) +
  geom_point(color="#9999CC",na.rm = F) +
  scale_x_datetime(labels = date_format("%y-%m"))+
  theme_bw()+
  labs(x=" ")+
  theme(axis.text=element_text(size=8))

P3=ggplot(data=cozie_train[,c("time",target_vars)], aes(x=time, y = q_noise_nearby)) +
  geom_point(color="#9999CC",na.rm = F) +
  scale_x_datetime(labels = date_format("%y-%m"))+
  theme_bw()+
  labs(x=" ")+
  theme(axis.text=element_text(size=8))

P4=ggplot(data=cozie_train[,c("time",target_vars)], aes(x=time, y = q_thermal_preference)) +
  geom_point(color="#9999CC",na.rm = F) +
  scale_x_datetime(labels = date_format("%y-%m"))+
```

```
  theme_bw()+
  labs(x=" ")+
  theme(axis.text=element_text(size=8))

ggarrange(P1,P2,P3,P4,nrow=4,ncol=1)
```




```
# plot(cozie_train$time,cozie_train[,target_vars[1]],
#      xlab="Time",ylab=target_vars[1])
# plot(cozie_train$time,cozie_train[,target_vars[2]],
#      xlab="Time",ylab=target_vars[2])
# plot(cozie_train$time,cozie_train[,target_vars[3]],
#      xlab="Time",ylab=target_vars[3])
# plot(cozie_train$time,cozie_train[,target_vars[4]],
#      xlab="Time",ylab=target_vars[4])
```

There is a time window between January and April with no observations. Let us split the data in two parts, one until 2023-01-21 15:00:16 (indexed 614909), and the other starting from 2023-03-27 11:55:01 (indexed 615204).

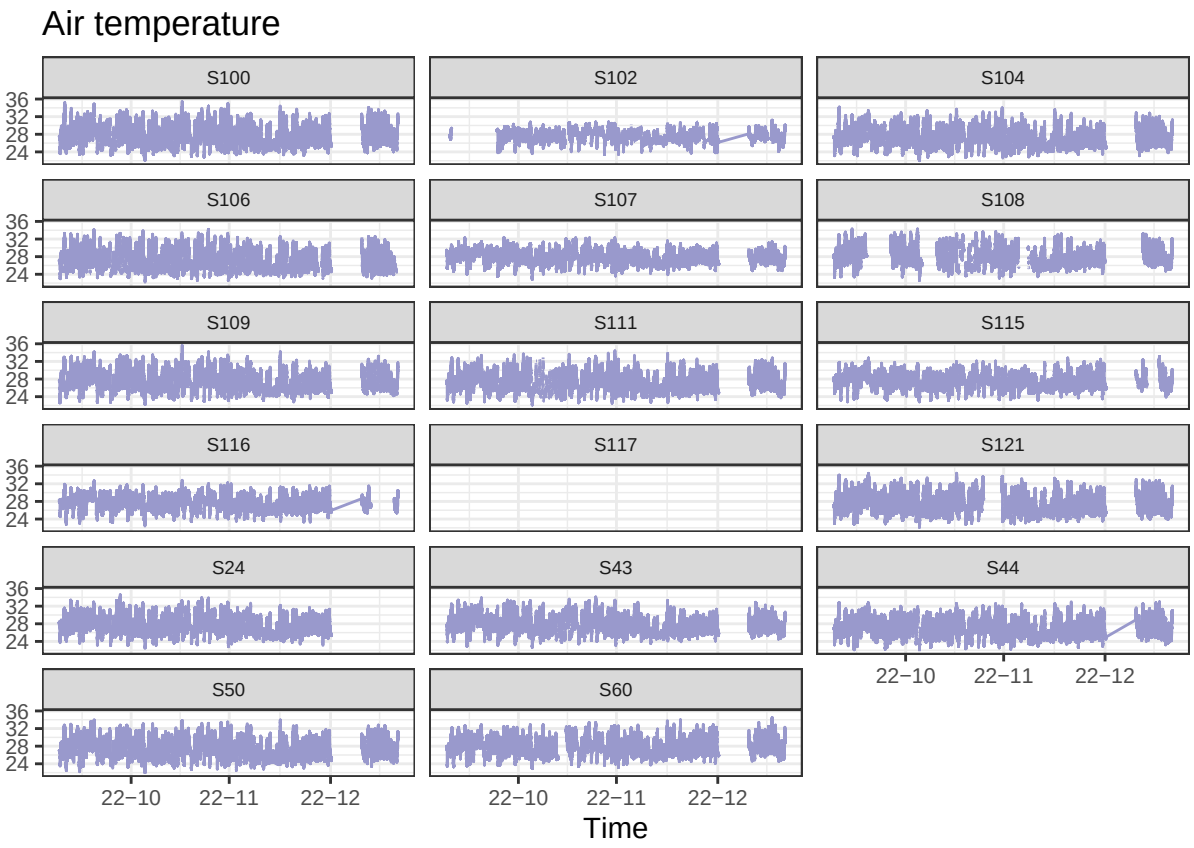
```
last_date_first_wind=cozie_train$time[614909]
first_date_second_wind=cozie_train$time[615204]
```

Weather variables, first window

```
air_temp1=air_temp[air_temp$time<=last_date_first_wind,]
rainfall1=rainfall[rainfall$time<=last_date_first_wind,]
RH1=RH[RH$time<=last_date_first_wind,]
wind_dir1=wind_dir[wind_dir$time<=last_date_first_wind,]
wind_speed1=wind_speed[wind_speed$time<=last_date_first_wind,]
```

Air temperature

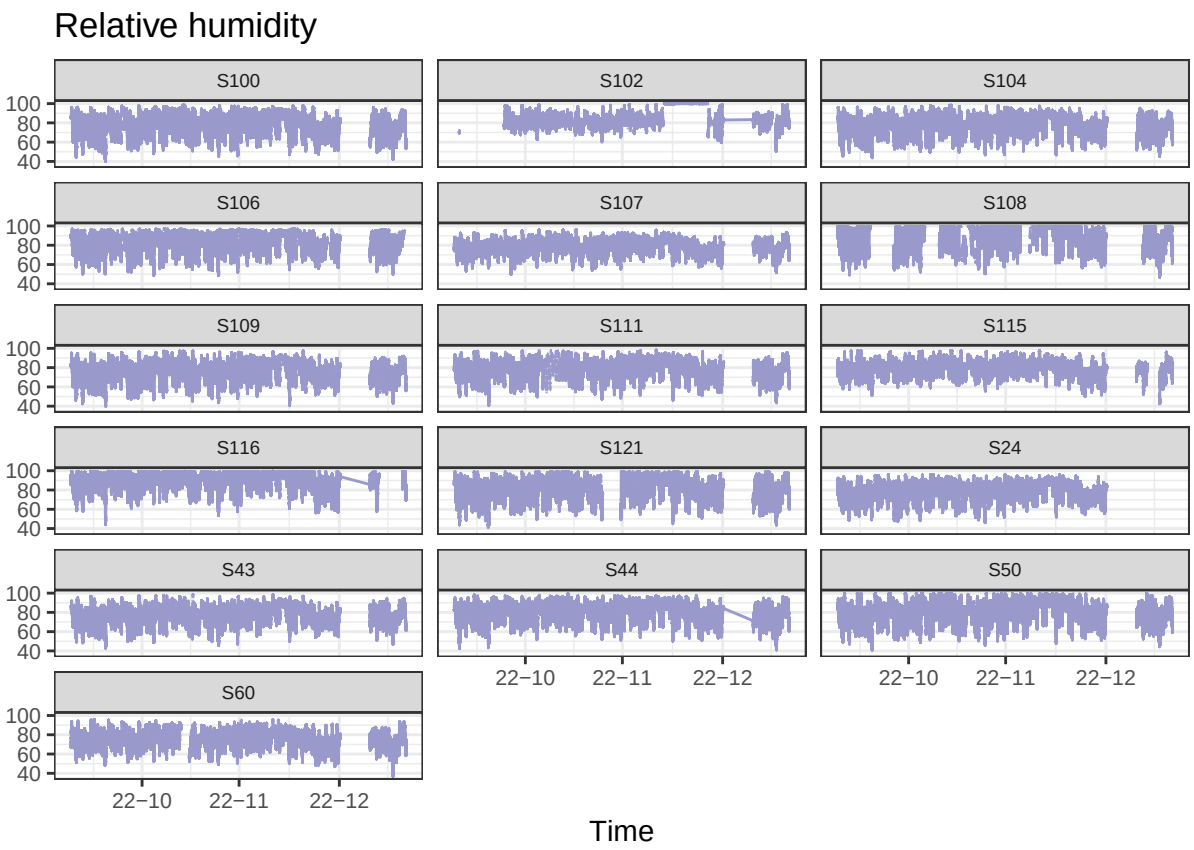
```
air_temp1_long=air_temp1 %>%
  pivot_longer(!time, names_to = "location", values_to = "air_temperature")

ggplot(data=air_temp1_long, aes(x=time, y = air_temperature)) +
  geom_line(color="#9999CC",na.rm = F) + facet_wrap(~location,nrow=6,ncol=3)+
  scale_x_datetime(labels = date_format("%y-%m"))+
  theme_bw()+
  labs(title="Air temperature",y=" ",x="Time")+
  theme(axis.text=element_text(size=8),strip.text.x = element_text(size = 7))
```



Relative humidity

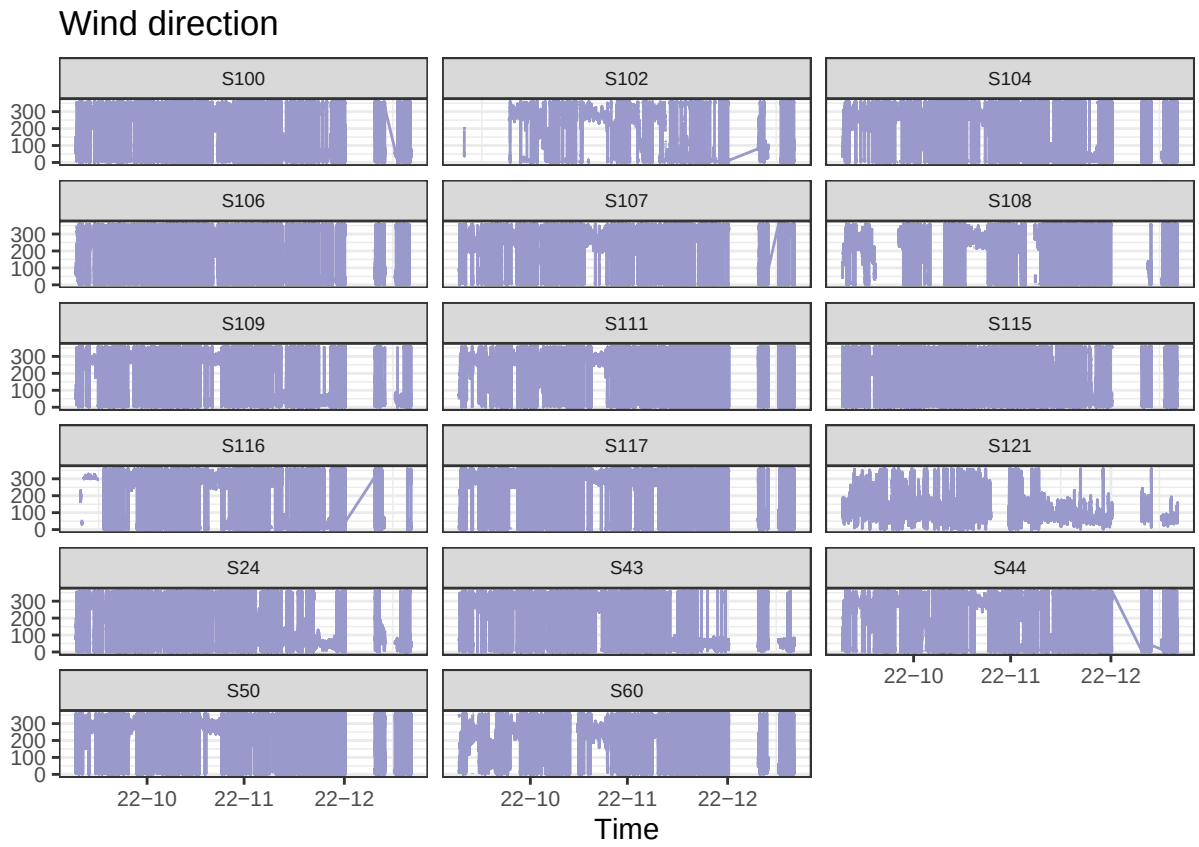
```
# Relative humidity
RH1_long=RH1 %>%
```

```
  pivot_longer(!time, names_to = "location", values_to = "rh")

ggplot(data=RH1_long, aes(x=time, y = rh)) +
  geom_line(color="#9999CC",na.rm = F) + facet_wrap(~location,nrow=6,ncol=3)+
  scale_x_datetime(labels = date_format("%y-%m"))+
  theme_bw()+
  labs(title="Relative humidity",y=" ",x="Time")+
  theme(axis.text=element_text(size=8),strip.text.x = element_text(size = 7))
```



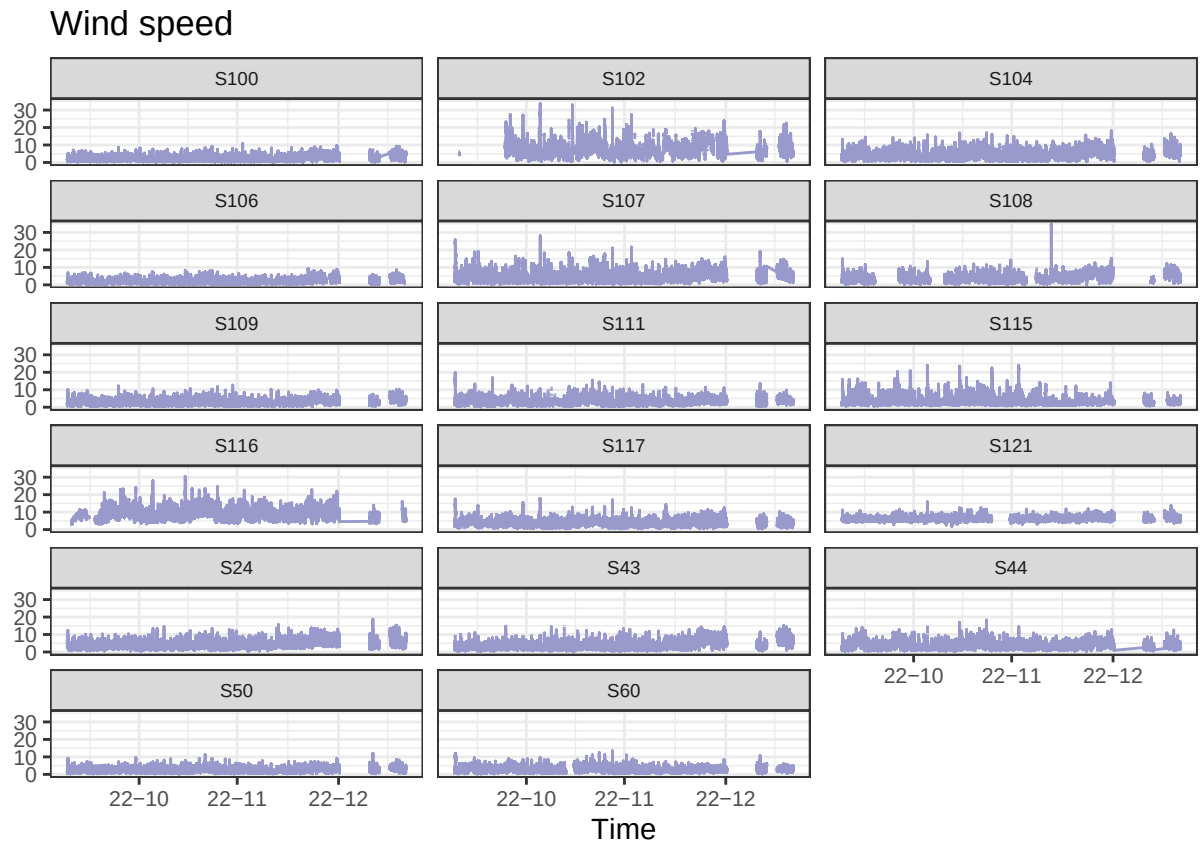
Wind direction

```
# Relative humidity
wind_dir1_long=wind_dir1 %>%
  pivot_longer(!time, names_to = "location", values_to = "wind_dir")

ggplot(data=wind_dir1_long, aes(x=time, y = wind_dir)) +
  geom_line(color="#9999CC",na.rm = F) + facet_wrap(~location,nrow=6,ncol=3)+
  scale_x_datetime(labels = date_format("%y-%m"))+
  theme_bw()+
  labs(title="Wind direction",y=" ",x="Time")+
  theme(axis.text=element_text(size=8),strip.text.x = element_text(size = 7))
```



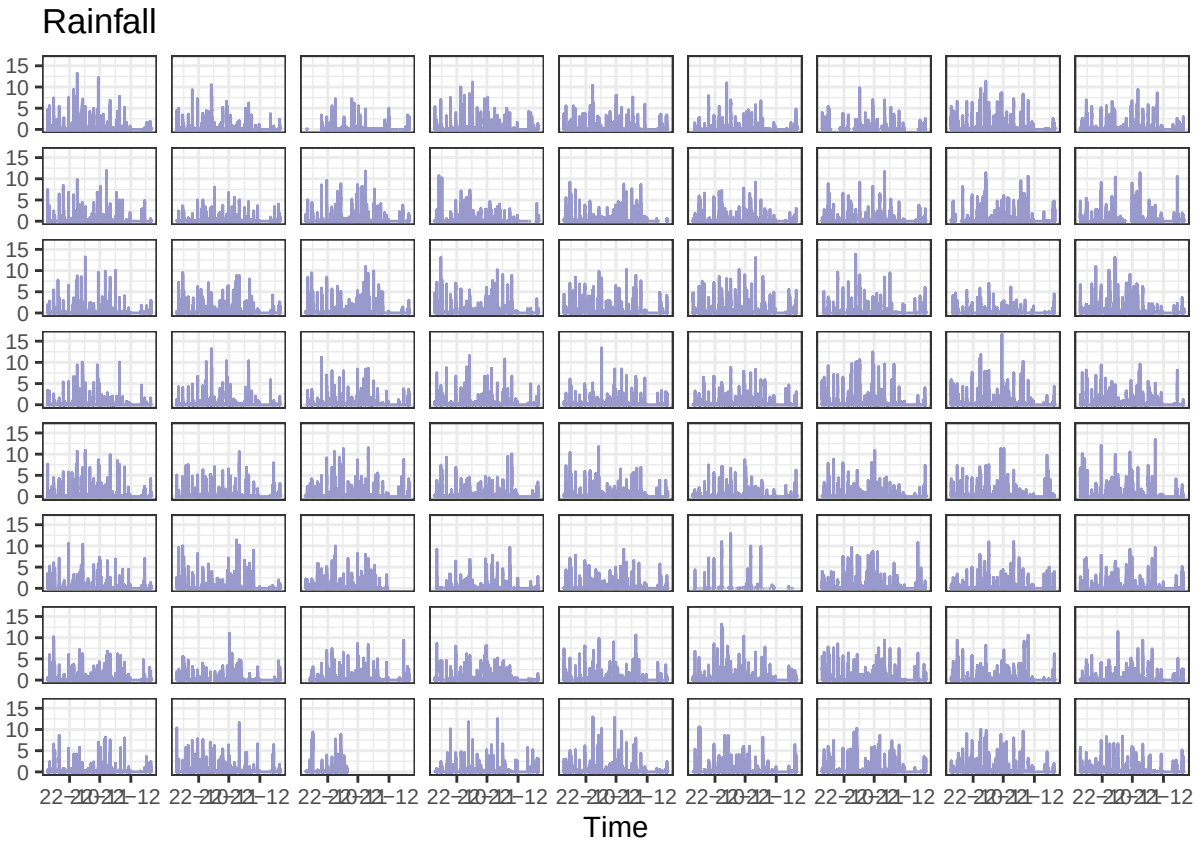
Wind speed

```
# Relative humidity
wind_speed1_long=wind_speed1 %>%
  pivot_longer(!time, names_to = "location", values_to = "wind_speed")

ggplot(data=wind_speed1_long, aes(x=time, y = wind_speed)) +
  geom_line(color="#9999CC",na.rm = F) + facet_wrap(~location,nrow=6,ncol=3)+
  scale_x_datetime(labels = date_format("%y-%m"))+
  theme_bw()+
  labs(title="Wind speed",y=" ",x="Time")+
  theme(axis.text=element_text(size=8),strip.text.x = element_text(size = 7))
```



Rainfall

```
# Relative humidity
rainfall1_long=rainfall1 %>%
  pivot_longer(!time, names_to = "location", values_to = "rf")

ggplot(data=rainfall1_long, aes(x=time, y = rf)) +
  geom_line(color="#9999CC",na.rm = F) + facet_wrap(~location)+
  scale_x_datetime(labels = date_format("%y-%m"))+
  theme_bw()+
  labs(title="Rainfall",y=" ",x="Time")+
  theme(axis.text=element_text(size=8),strip.background=element_blank(),  strip.text.x = element_blank()
```


Rainfall
15
10
5
0
Time