3_deskriptivne_analize

May 13, 2025

[1]:
```
%load_ext autoreload
%autoreload 2
```

[ ]:
```
import pandas as pd
import locale
import numpy as np

from scipy.stats import permutation_test,mannwhitneyu
from helpers import filter_region_year, split_directive
```

[3]:
```
NEVARNE_SNOVI = ["SO2", "PM10", "NO2"]
locale.setlocale(locale.LC_ALL, "sl_SI.utf8")

df_urne = pd.read_csv("podatki/df_urne.csv", parse_dates=["Datum"]).drop(
    columns="Postaja"
)
df_dnevne = pd.read_csv("podatki/df_dnevne.csv", parse_dates=["Datum"]).drop(
    columns="Postaja"
)
df_mesecne = pd.read_csv("podatki/df_mesecne.csv", parse_dates=["Datum"]).drop(
    columns="Postaja"
)
```

[ ]:
```
def get_significance(p):
    return "*" * sum(p < t for t in [0.05, 0.01, 0.001])


def format_p(p):
    return "<0.001***" if p < 0.001 else f"{p:.3g}{get_significance(p)}"


def create_combined_stats_table(df_urne, df_dnevne, df_mesecne, pollutant):
    """
    Combines statistics, performs permutation and Mann-Whitney U tests.

    Args:
        df_urne (pd.DataFrame): DataFrame with hourly data.
        df_dnevne (pd.DataFrame): DataFrame with daily data.
```

```
        df_mesecne (DataFrame): DataFrame with monthly data.
        pollutant (str): The pollutant to analyze.

    Returns:
        Tuple[pd.DataFrame, pd.DataFrame, pd.DataFrame]: (combined_df, perm_df,␣
↪mw_df).
    """

    dfs = {
        "(najvišje) urne meritve": df_urne,
        "(najvišje) dnevne meritve": df_dnevne,
        "mesecne meritve": df_mesecne,
    }
    dfs = {k: v for k, v in dfs.items() if v is not None and v is not None}

    regions = set(reg for df in dfs.values() for reg in df["Regija"].unique())
    regions = sorted(list(regions))

    basic_columns = [
        (freq, label)
        for freq in dfs.keys()
        for label in [
            "N [pred]",
            "N [po]",
            "Povprečje [pred]",
            "Povprečje [po]",
            "Mediana [pred]",
            "Mediana [po]",
        ]
    ]

    mw_columns = [
        (freq, "MW test p-vrednost") for freq in dfs.keys()
    ]  # Mann Whitney U Test

    multi_cols = pd.MultiIndex.from_tuples(basic_columns)
    combined_df = pd.DataFrame(index=regions, columns=multi_cols)
    combined_df = combined_df.sort_index()

    mw_multi_cols = pd.MultiIndex.from_tuples(mw_columns)
    mw_df = pd.DataFrame(index=regions, columns=mw_multi_cols)
    mw_df = mw_df.sort_index()

    def assign_permutation(
        df: pd.DataFrame,
    ):
        before, after = split_directive(df, pollutant)
```

```
        before_data = before[pollutant].dropna().to_numpy()
        after_data = after[pollutant].dropna().to_numpy()

        if len(before_data) == 0 or len(after_data) == 0:
            return np.nan

        num_rounds = 10_000

        result = permutation_test(
            data=(before_data, after_data),  # Pass in numpy arrays
            statistic=lambda x, y: np.mean(x) - np.mean(y),
            permutation_type="independent",
            alternative="greater",  # test if before is greater
            n_resamples=num_rounds,
            vectorized=False,
        )
        return result.pvalue



    def assign_mw(df: pd.DataFrame, mw_df: pd.DataFrame, region: str, freq:␣
↪str):
        before, after = split_directive(df, pollutant)

        before_data = before[pollutant].dropna()
        after_data = after[pollutant].dropna()
        # Mann Whitney U Test needs minimum of 1 value to be valid
        if (len(before_data) < 1) or (len(after_data) < 1):
            mw_df.loc[region, (freq, "MW test p-vrednost")] = np.nan
            return mw_df
        try:
            u_statistic, p_value = mannwhitneyu(
                before_data, after_data, alternative="greater"
            )
            mw_df.loc[region, (freq, "MW test p-vrednost")] = format_p(p_value)

        except Exception as e:
            print(f"Mann Whitney Test Error: {e}")
            mw_df.loc[region, (freq, "MW test p-vrednost")] = np.nan

        return mw_df

    def get_basic_stats(
        df: pd.DataFrame, combined_df: pd.DataFrame, region: str, freq: str
    ):
        before, after = split_directive(df, pollutant)
```

```
        n_pred = before[pollutant].count()
        n_po = after[pollutant].count()
        povprecje_pred = before[pollutant].mean()
        povprecje_po = after[pollutant].mean()
        mediana_pred = before[pollutant].median()
        mediana_po = after[pollutant].median()

        combined_df.loc[region, (freq, "N [pred]")] = n_pred
        combined_df.loc[region, (freq, "N [po]")] = n_po
        combined_df.loc[region, (freq, "Povprečje [pred]")] = povprecje_pred
        combined_df.loc[region, (freq, "Povprečje [po]")] = povprecje_po
        combined_df.loc[region, (freq, "Mediana [pred]")] = mediana_pred
        combined_df.loc[region, (freq, "Mediana [po]")] = mediana_po

        return combined_df

    for freq, df in dfs.items():
        if df is not None and pollutant in df.columns and "Regija" in df.
 ↪columns:
            try:
                for region in regions:
                    region_df = df[df["Regija"] == region].copy()  # filter by␣
 ↪regija
                    combined_df = get_basic_stats(region_df, combined_df,␣
 ↪region, freq)
                    mw_df = assign_mw(region_df, mw_df, region, freq)

            except Exception as e:
                print(f"Error processing {freq} for pollutant {pollutant}: {e}")

    return (combined_df.apply(pd.to_numeric, errors="coerce").round(1), mw_df)
```

[5]:
```
combined_df, so2_perm,mw_so2 = create_combined_stats_table(
    filter_region_year(df_urne, "SO2"),
    filter_region_year(df_dnevne, "SO2"),
    filter_region_year(df_mesecne, "SO2"),
    "SO2",
)

display(so2_perm)
display(mw_so2)
```

	(najvišje) urne meritve	(najvišje) dnevne meritve
	perm. test p-vrednost	perm. test p-vrednost
Koroška	<0.001***	<0.001***
Osrednjeslovenska	<0.001***	<0.001***
Posavska	<0.001***	<0.001***

```
Savinjska                        <0.001***                 <0.001***
Zasavska                         <0.001***                 <0.001***
```

```
                        mesecne meritve
                   perm. test p-vrednost
Koroška                         <0.001***
Osrednjeslovenska               <0.001***
Posavska                        <0.001***
Savinjska                       <0.001***
Zasavska                        <0.001***
```

```
                   (najvišje) urne meritve (najvišje) dnevne meritve  \
                        MW test p-vrednost        MW test p-vrednost
Koroška                          <0.001***                 <0.001***
Osrednjeslovenska                <0.001***                 <0.001***
Posavska                         <0.001***                 <0.001***
Savinjska                        <0.001***                 <0.001***
Zasavska                         <0.001***                 <0.001***
```

```
                      mesecne meritve
                   MW test p-vrednost
Koroška                         <0.001***
Osrednjeslovenska               <0.001***
Posavska                        <0.001***
Savinjska                       <0.001***
Zasavska                        <0.001***
```

[6]:
```
combined_pm10_table, pm10_perm,mw_pm10 = create_combined_stats_table(
    filter_region_year(df_urne, "PM10"),
    filter_region_year(df_dnevne, "PM10"),
    filter_region_year(df_mesecne, "PM10"),
    "PM10",
)

display(pm10_perm)
display(mw_pm10)
```

```
                        mesecne meritve
                   perm. test p-vrednost
Goriška                         <0.001***
Osrednjeslovenska               <0.001***
Podravska                       <0.001***
Pomurska                        <0.001***
Posavska                         0.0052**
Savinjska                       <0.001***
Zasavska                        <0.001***
```

```
                      mesecne meritve
                   MW test p-vrednost
```

```
Goriška                    <0.001***
Osrednjeslovenska          <0.001***
Podravska                  <0.001***
Pomurska                   <0.001***
Posavska                   <0.001***
Savinjska                  <0.001***
Zasavska                   <0.001***
```

[7]:
```
combined_no2_table,no2_perm,mw_no2 = create_combined_stats_table(
    filter_region_year(df_urne, "NO2"),
    filter_region_year(df_dnevne, "NO2"),
    filter_region_year(df_mesecne, "NO2"),
    "NO2",
)

display(no2_perm)
display(mw_no2)
```

```
                 (najvišje) urne meritve       mesecne meritve
                   perm. test p-vrednost perm. test p-vrednost
Goriška                             0.625              0.0095**
Osrednjeslovenska                       1                     1
Podravska                      <0.001***             <0.001***
Pomurska                       <0.001***                0.0351*
Savinjska                      <0.001***                  0.139
Zasavska                       <0.001***              0.0011**
```

```
                 (najvišje) urne meritve    mesecne meritve
                      MW test p-vrednost MW test p-vrednost
Goriška                              0.62          0.00319**
Osrednjeslovenska                       1                  1
Podravska                      <0.001***          <0.001***
Pomurska                       <0.001***             0.0235*
Savinjska                      <0.001***               0.965
Zasavska                       <0.001***          0.00264**
```