Aufgabe 1: Häufigkeitsverteilung (2 Punkte)

Folgende Werte sind gegeben: 0.1, 1.2, 0.4, 0.3, 1.7, 2.5, 0.6 und 1.3. Stellen Sie die Häufigkeitsverteilung und die relative Häufigkeitsverteilung für Intervalle der Länge 1 im Wertebereich von 0 bis 3 grafisch dar. Eine grobe Skizze reicht aus.

Aufgabe 2: Standardabweichung (12 Punkte)

a) Berechnen Sie monatlich gemittelte Temperaturen für Januar für Leipzig basierend auf dem Datensatz in Übungsblatt 1, Aufgabe 4. Berücksichtigen Sie nur Monate, für die an allen Tagen eine Tagesmitteltemperatur ermittelt wurde. Stellen Sie die gemittelten Temperaturen für Januar als Zeitreihe dar. Unten finden Sie Vorschläge zur Lösung dieser Aufgabe (optional). (4 Punkte)

b) Berechnen Sie die Standardabweichung der Januar-Mitteltemperaturen für die Jahre ab 1950. Schreiben Sie dazu selbst eine Funktion und rufen Sie diese Funktion auf. Benutzen Sie hierzu bitte keine vorhandene Funktion aus einem Python-Paket. (8 Punkte)

Aufgabe 3: Relative Häufigkeitsverteilung (10 Punkte)

a) Stellen Sie die relative Häufigkeitsverteilung für die Januar-Monatsmitteltemperaturen für die Jahre ab 1950 für 2.5°C Intervalle von -12.5°C bis 10°C graphisch dar. Berücksichtigen Sie nur Monate, für die an allen Tagen ein gültiger Wert vorliegt. Benutzen Sie eine vorhandene Python Funktion (`plt.hist`). Sie können hierzu gerne einfach den Programmcode aus Aufgabe 2 erweitern. (4 Punkte)

b) Stellen Sie die zugehörige kumulierte relative Häufigkeitsverteilung graphisch dar. (1 Punkt)

c) Stellen Sie die relative Häufigkeitsverteilung der Niederschlagshöhe für Januar graphisch dar. Wählen Sie geeignete Intervalle. (2 Punkte)

d) Wiederholen Sie Teil a für die tägliche (nicht monatliche!) Niederschlagshöhe für Januar. Wählen Sie geeignete Intervalle. Ist die Schiefe der Verteilung positiv oder negativ? Tipp: Benutzen Sie eine logarithmische y-Achse. (3 Punkte)

Hinweis für Python: `https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.hist.html` entweder mit `weights=np.ones_like(x)/float(len(x))` oder falls Sie Pandas verwenden, und Sie fehlende Werte haben, mit `weights=np.ones_like(x)/x.count()`. Bei `x.count()` werden fehlende Werte (NaN) nicht mitgezählt.

Vorschläge zur Lösung von Augabe 2a mit Python (optional):

1. Mit Pandas

- Importieren Sie `pandas` als `pd` wie im Programm `valid_with_pandas.py` aus der ersten Übung. Importieren Sie außerdem `numpy` als `np`.

- Nutzen Sie wieder `read_csv` zum Einlesen der Daten, also z. B.:
  `ds = pd.read_csv(ifile, delimiter=';', encoding='iso-8859-1')`
  wobei `ifile` der Dateiname ist.
- Konvertieren Sie die Datumswerte (integer im Format `%Y%m%d`) in der Spalte `MESS_DATUM` mit Pandas `to_datetime` in `Datetime` Objekte:
  `ds['MESS_DATUM']=pd.to_datetime(ds['MESS_DATUM'], format='%Y%m%d')`
- Nutzen Sie Pandas `replace` um fehlende `TMK`-Werte durch `np.nan` zu ersetzen (nan: „Not a number“).
- Gruppieren Sie die Daten nach Monaten und berechnen Sie Mittelwerte.

  Definieren Sie dazu zunächst eine Gruppe mit Pandas `Grouper`. Geben Sie `'MESS_DATUM'` als Schlüssel (`key`) an. Geben Sie außerdem die Frequenz als optionales Argument an (siehe Link unter `freq` und Beispiele in der Dokumentation zu `Grouper`).

  Die Anweisung, um eine Gruppe zu erzeugen, lautet `pd.Grouper(key=..., freq=...)`, wobei Sie die ... noch ersetzen müssen.

  Die Gruppierung erfolgt mithilfe der Anweisung `groupby`. Um den Mittelwert zu bilden, könnten Sie die Methode `mean` verwenden. Allerdings werden dabei nan-Werte teilweise einfach übersprungen und die Lösung wird falsch. Auch das optionale Argument `skipna=False` funktioniert im Moment bei `DataFrameGroupBy`-Objekten nicht. Sie können das Problem umgehen, indem Sie mit `agg(lambda x: x.mean(skipna=False)` mitteln. Am Ende könnte der Befehl ungefähr so aussehen:

  ```
  ds=ds.groupby(GRUPPE).agg(lambda x:  x.mean(skipna=False))
  ```

  Dabei müssten Sie allerdings noch `GRUPPE` durch `pd.Grouper(...)` ersetzen.
- Erzeugen Sie einen neuen Index: `ds.reset_index(inplace=True)`
- Wählen Sie den Monat Januar aus: `ds=ds[ds['MESS_DATUM'].dt.month == 1]`
- Sie können nun `ds[' TMK']` gegen `ds['MESS_DATUM']` auftragen.
- Zur korrekten Darstellung müssen Sie nun noch eine durchgehende Zeitachse definieren und fehlende Zeiten im Datensatz durch mit fehlenden Werten (z. B, `np.nan`) auffüllen.

2. Ohne Pandas

- Beginnen Sie mit dem einer Kopie des Programms `valid_without_pandas.py` aus der ersten Übung
- Im dem Programm wird ein ASCII-Datensatz zeilenweise eingelesen. Definieren Sie analog zur Temperatur (`tmk`) eine weitere Variable, welche das Datum aus dem Datensatz enthält.
- Lassen Sie sich die Werte für das Datum zunächst einfach nur auf den Bildschirm ausgeben. Bestimmen Sie nun den Monat. Dazu können Sie (siehe Folien zur ersten Übung) z. B. das Datum zunächst (falls nötig) in einen String umwandeln. Aus dem

String können Sie den Monat extrahieren. Diesen neuen String können Sie dann wieder in einen Integer umwandeln. Alternativ könnten Sie das Datum in einen Float umwandeln und den Monat durch Dividieren und Subtrahieren berechnen.

- Bestimmen Sie außer dem Monat am besten auch gleich noch das Jahr.
- Lassen Sie sich lediglich das tägliche Datum und die tägliche Temperatur für Januar mithilfe einer `if`-Anweisung auf den Bildschirm ausgeben.
- Definieren Sie eine neue Variable (z. B. `msum`) vom Typ Float, und zwar oberhalb der Einlese-Schleife. Setzen Sie diese Variable zunächst auf Null. Dazu könnten Sie z. B. direkt unterhalb der Zeile `i_valid=0` eine neue Zeile mit `msum=0.` einfügen.
- Berechnen Sie zunächst die Summe über alle Januar-Temperaturen, indem Sie `tmk` innerhalb der Schleife zu `msum` hinzuaddieren. Die neue Zeile lautet: `msum=msum+tmk`.
- Bestimmen Sie mithilfe einer `if`-Anweisung innerhalb der Einlese-Schleife, wenn sich das Jahr ändert.
- Überprüfen Sie innerhalb des `if`-Blocks, ob für den vorigen Januar an allen Tagen eine Tagesmitteltemperatur vorliegt.
- Berechnen Sie ggf. aus `msum` einen Monatsmittelwert (`mavg`). Geben Sie diese Mittelwerte zunächst auf den Bildschirm aus.
- Erstellen Sie oberhalb der Einlese-Schleife eine leere Liste, z. B. mit `mmlist=[]`. Mit `mmlist.append(mavg)` können Sie nun einen einzelnen Mittelwert an diese Liste anhängen.
- Falls nicht genügend Tagesmitteltemperaturen vorliegen (oder falls ganze Jahre fehlen), können Sie eventuell ein `None` in die Liste einfügen.
- Stellen Sie die Werte in der Liste grafisch dar.