

Statistik für Data Scientists

Vorlesung 3: Robuste Verteilungsdiagnostik: Histogramm, KDE,
ECDF/QQ

Prof. Dr. Siegfried Handschuh
DS-NLP
Universität St. Gallen

Agenda

1. Aussreißer erkennen und behandeln (fortgesetzt)
2. Histogramm und **KDE** (Kernel Density Estimation)
3. eBox und Violin für Gruppen
4. **ECDF** (Empirical Cumulative Distribution Function) und **QQ** (Quantile-Quantile-Plot) für Formdiagnostik

Letzte Woche

1. Lagekennzahlen: Mittelwert, Median, Modus, Quantile
2. Streuungskennzahlen: Varianz & SD, IQR, MAD
3. Ausreisser erkennen und behandeln (angefangen): Z-Score, Tukey-Fences, Mod. Z-Score

Ausreißer erkennen und behandeln

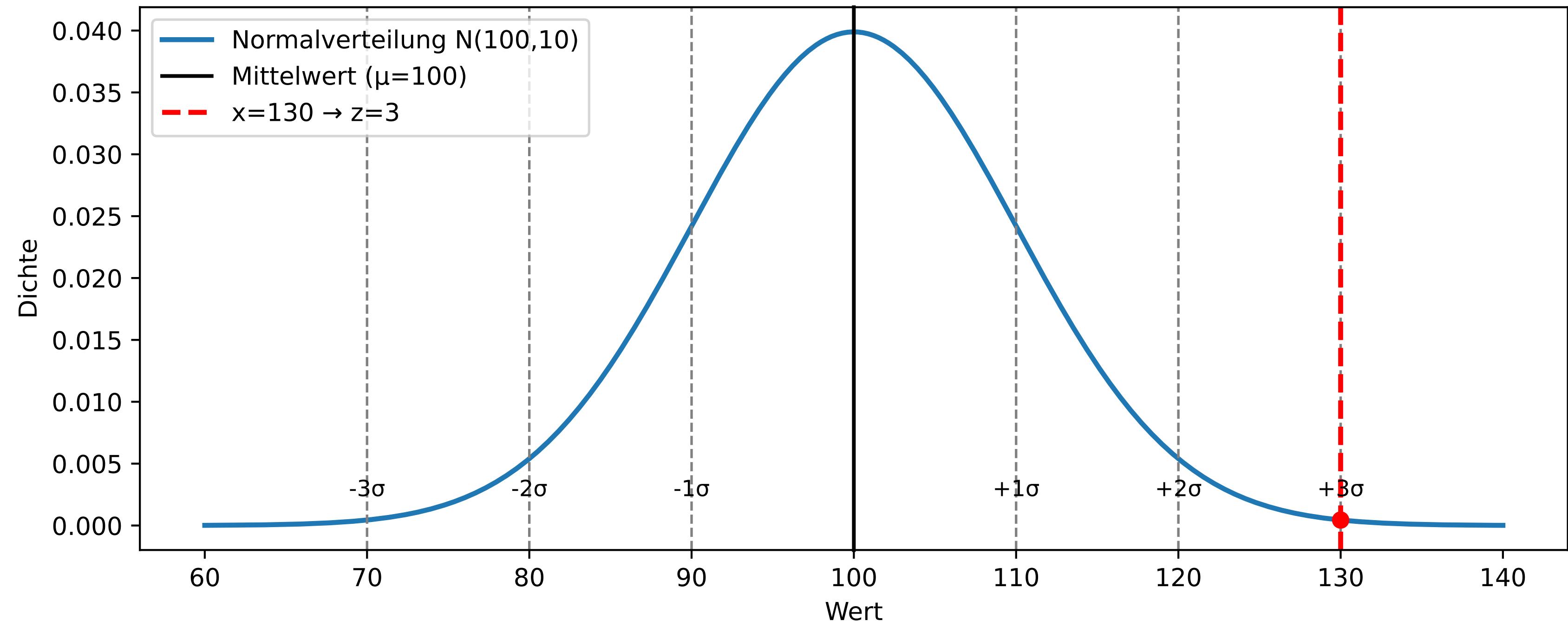
Drei Ansätze zur Ausreißer-Erkennung

Verfahren	Basis	Eignung
Klassischer Z-Score	Mittelwert & Standardabw.	sinnvoll nur bei (fast) normalverteilten Daten
Tukey-Fences (Boxplot)	Quartile & IQR	robust, keine Verteilungsannahme
Modifizierter Z-Score	Median & MAD	sehr robust, geeignet bei Schiefe & Heavy Tails

Klassischer Z-Score

- Misst, wie viele Standardabweichungen ein Wert vom Mittelwert entfernt ist
 - **Definition:** $z_i = \frac{x_i - \bar{x}}{s}$
 - **Annahme:** Daten sind (nahezu) normalverteilt
 - **Typische Schwelle:** $|z| > 3 \rightarrow$ Ausreißerkandidat
- 👉 Nützlich bei Normalverteilung, empfindlich bei Schiefe und Ausreissern.

Z-Score Beispiel: $x=130, \mu=100, s=10 \rightarrow z=3$



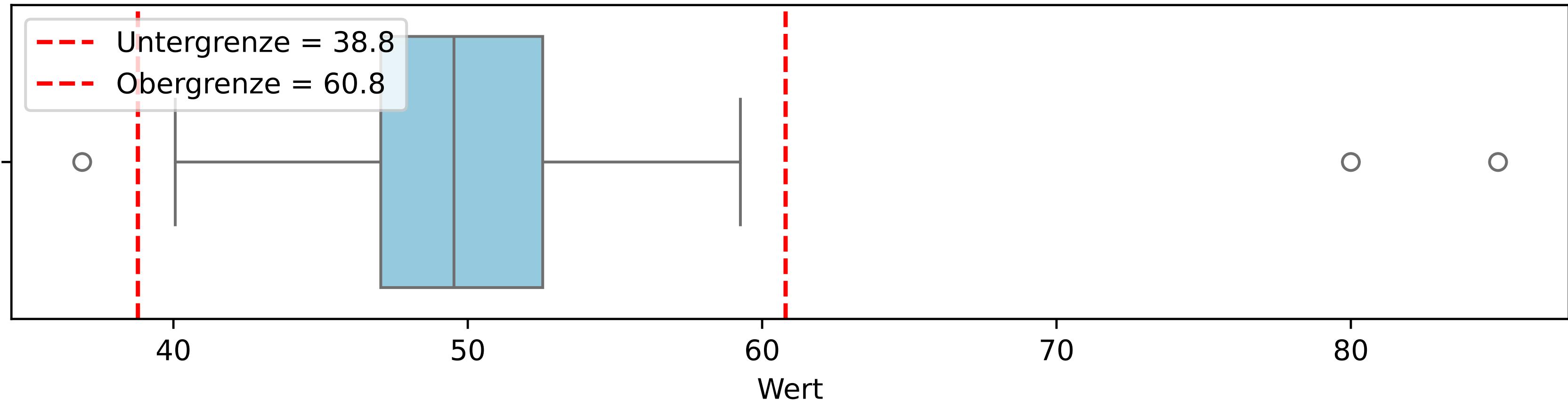
Tukey Fences mit Quartilen

IQR-Regel liefert robuste Kandidaten.

- $IQR = Q_3 - Q_1$
- Untere Grenze $Q_1 - 1.5 \cdot IQR$
- Obere Grenze $Q_3 + 1.5 \cdot IQR$
- Identifiziert Kandidaten, ohne Verteilungsannahme

Mini-Check: Was passiert bei breiterer Verteilung?

Boxplot mit Tukey-Fences ($1.5 \cdot \text{IQR}$)



Modifizierter Z-Score mit MAD

Robuste Lage und Streuung kombiniert.

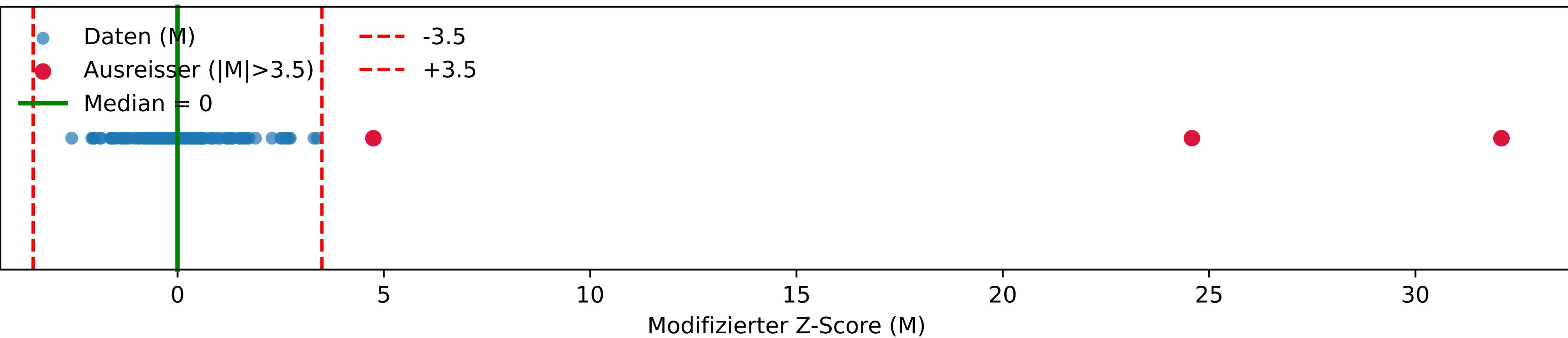
- $\text{MAD} = \text{median}(|x - \tilde{x}|)$
- $M_i = 0.6745 \cdot \frac{x_i - \tilde{x}}{\text{MAD}}$ → lineare Transformation
- **Schwelle:** $|M_i| > 3.5 \rightarrow \text{Ausreisser}$
- Gut bei Schiefe

Mini-Check: Warum MAD stabiler als SD?

Median + MAD-Band (Rohwert-Skala)



Schwellen ± 3.5 (M-Skala)



Einfluss von Ausreisern

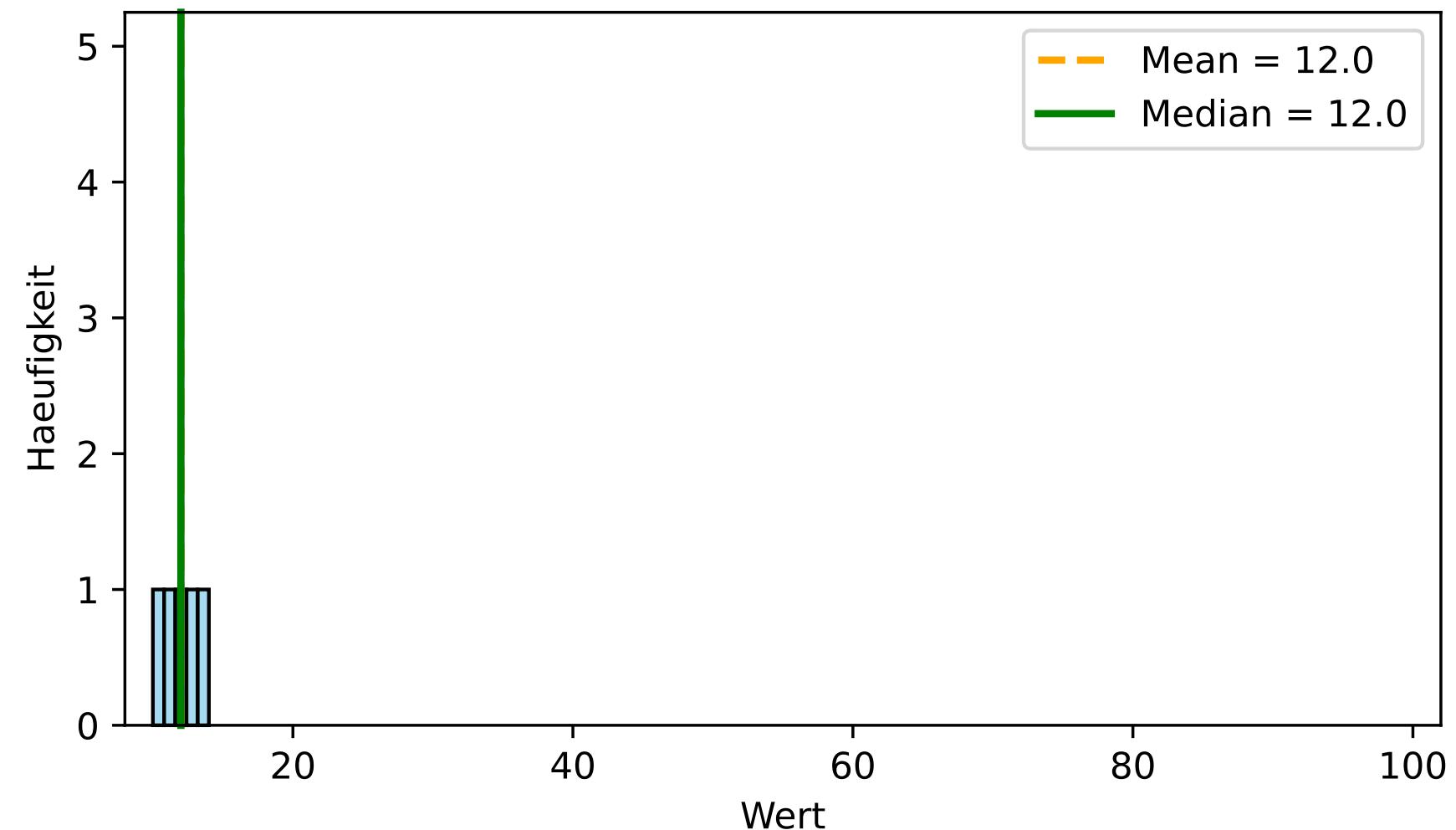
Einzelne Extremwerte ziehen Mean & SD stark nach oben

- Beispiel [10, 11, 12, 13, 14, 100]
- \bar{x} steigt stark durch den Ausreißer
- s (Standardabweichung) bläht sich auf
- Median & IQR verändern sich nur moderat
- Vor Modellierung prüfen

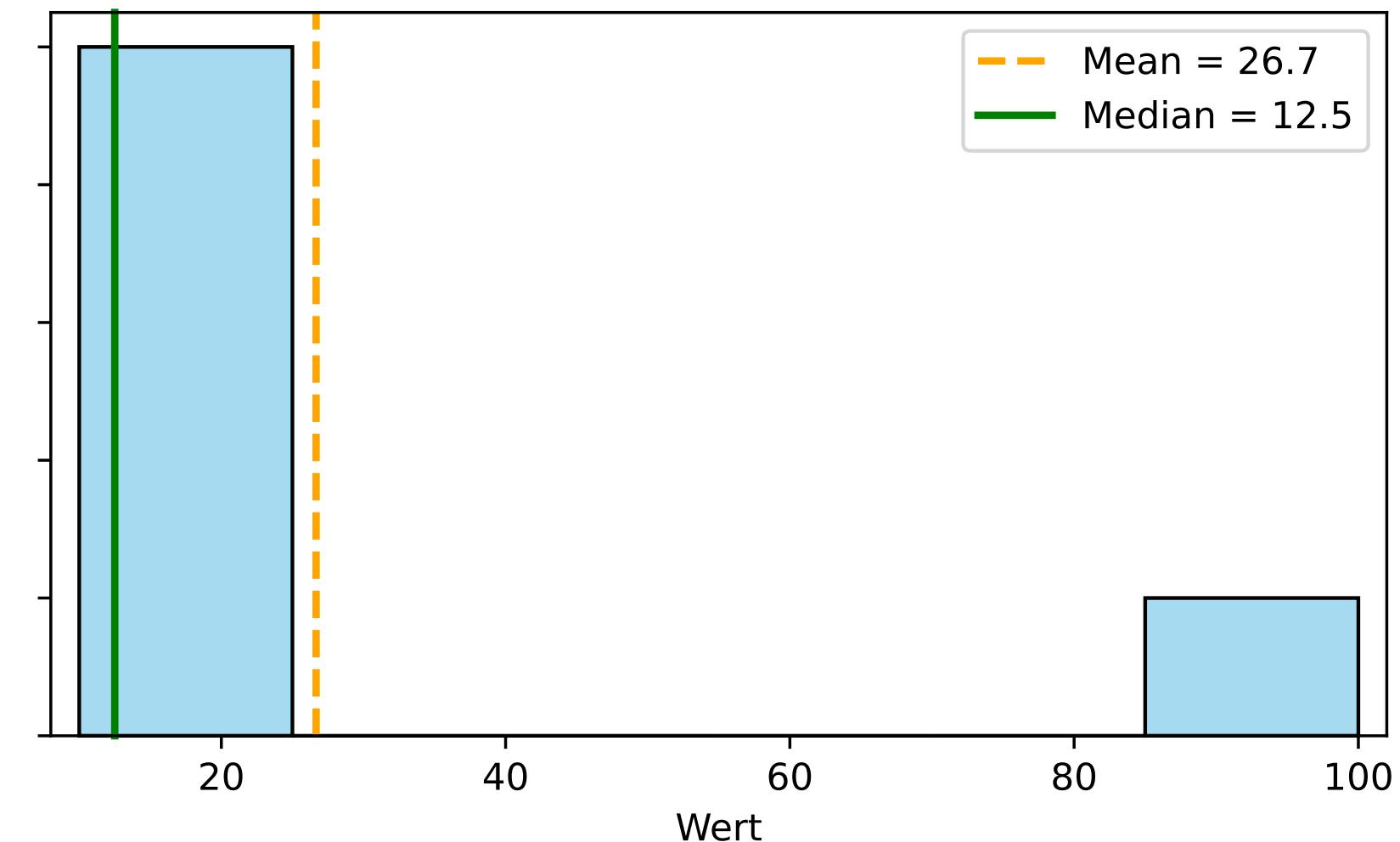
Mini-Check: Welche Kennzahl kippt zuerst und warum?

Ein Ausreißer zieht Mean & SD stark - Median & IQR bleiben stabil

Ohne Ausreißer



Mit Ausreißer (100)



Strategien im Umgang

Markieren, winsorisieren, trimmen mit Begründung.

- Markieren und separat berichten
- Winsor: auf Grenze setzen (deckeln)
- Trimmen: Ränder entfernen
- Immer Entscheidung dokumentieren

Mini-Check: Domäne für Markieren statt Entfernen?

Markieren



Ausreißer markieren,
separat berichten

Winsor



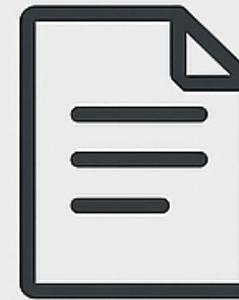
Werte auf
Grenze setzen

Trimmen



Ränder
entfernen

Dokumentieren



Entscheidung
festhalten

Python: Tukey und mod. Z

```
q1,q3 = s.quantile([.25,.75]); iqr = q3-q1  
lo,hi = q1-1.5*iqr, q3+1.5*iqr  
mask_tukey = (s<lo)|(s>hi)  
  
med = s.median(); mad = (s-med).abs().median()  
M = 0.6745*(s-med)/mad  
mask_modz = M.abs()>3.5
```

Tukey: 2 Ausreisser markiert
ModZ: 1 Ausreisser markiert

Fallnotiz und Reproduzierbarkeit

Entscheidungen schriftlich festhalten.

- Regel, Schwelle, Datum, Variable
- Maske und Datenversion speichern
- Varianten mit und ohne Ausreißer berichten
- Nutzen für Audit und Kollaboration

Mini-Check: Drei Metadaten für die Notiz?

Fallnotiz und Reproduzierbarkeit

Regel	Schwelle	Datum	Kommentar
mod. Z-Score	3,5	21.05.2025	Variable X markiert
IQR	$> Q3 + 1,5 * IQR$	15.06.2025	Nur in Variante B entfernt
Saubere Dokumentation erhöht Transparenz & Vertrauen.			

Dokumentation

Feld	Eintrag
Regel	Tukey $1.5 \text{ } IQR + \text{mod.}Z \text{ } 3.5$
Schwelle & Grund	Robuste Erkennung, da Schiefe vermutet
Datum & Version	Datenstand 2025-09-20, Codeversion v1.3
Variable	<code>flipper_length_mm</code>
Ergebnis	5 Kandidaten, 3 real, 2 Messfehler
Entscheid	Markieren, Bericht mit und ohne

Heavy Tails vs echte Ausreisser

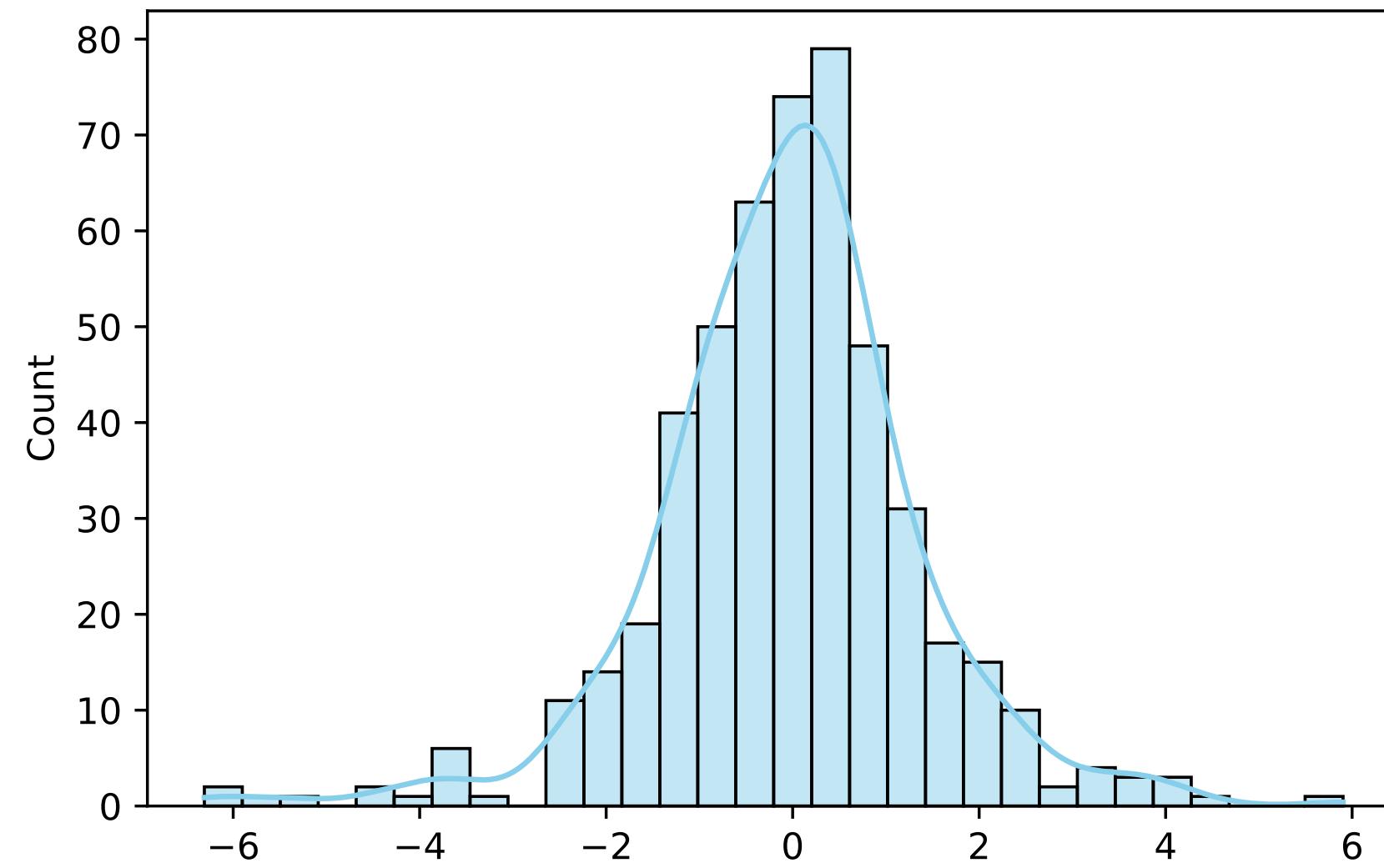
Heavy Tails (schwere Ausläufer) sind nicht automatisch Fehler.

- Heavy Tails = viele grosse Werte, nicht zwingend Fehler
- Echte Ausreisser = Messfehler, falsche Einheit, Datenproblem
- Diagnose mit Plots: Hist, KDE, ECDF, QQ
- Kontext berücksichtigen

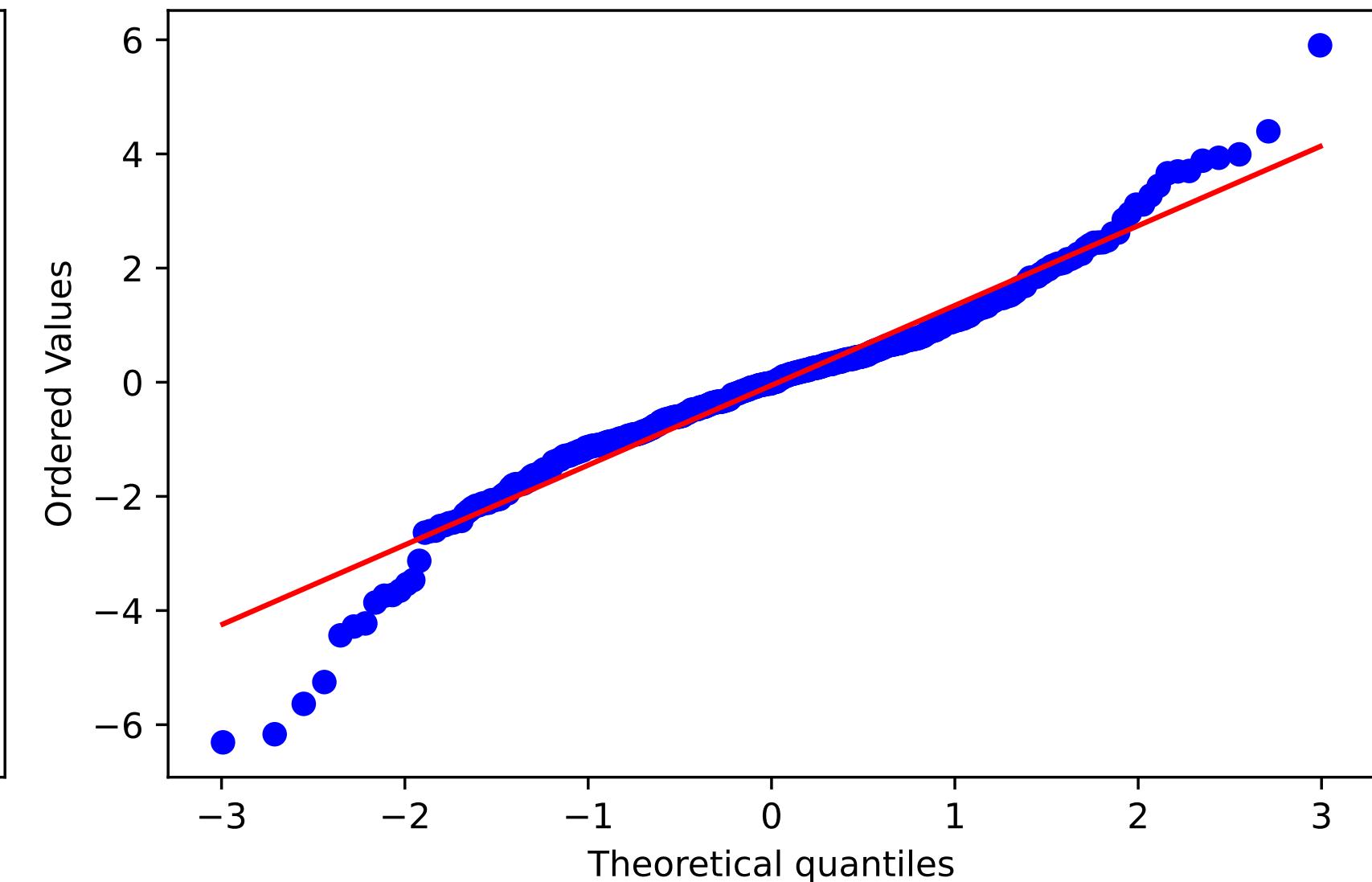
Mini-Check: Welcher Plot zeigt Tails klar?

👉 Nicht jeder Extremwert ist ein Fehler.

Histogramm + KDE (heavy tails)



QQ-Plot zeigt Abweichung in den Tails

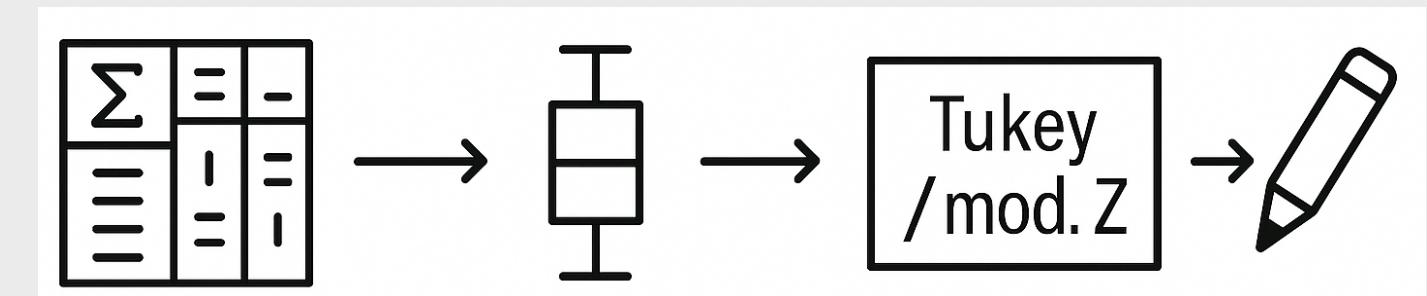


Pipeline: Kennzahl → Plot → Entscheid

Feste Reihenfolge verhindert Ad-hoc-Fehler.

1. Kennzahlen berechnen
2. Plots prüfen
3. Regel anwenden
(z. B. Tukey, mod. Z)
4. Entscheidung dokumentieren

Zuerst **Zählen**, dann **Schauen**, dann **Handeln**, dann **Aufschreiben**.

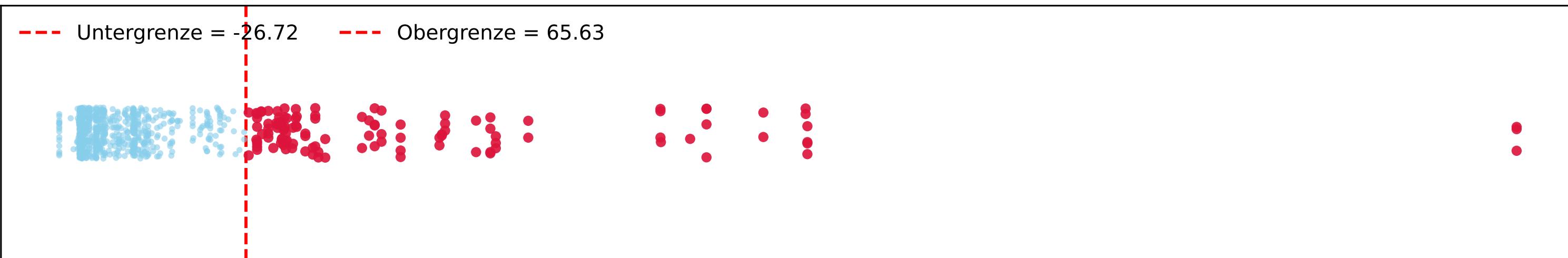


Mini-Check:
Warum erst robust dann Plots?

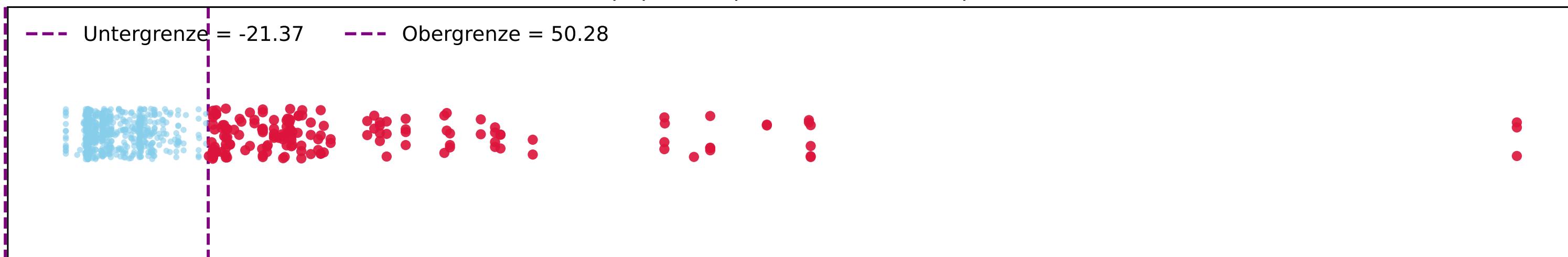
Mini-Aufgabe: Ausreißer finden

- Datensatz: Penguins flipper_length_mm
 - Wende Tukey und mod. Z an
 - Vergleiche Ergebnisse
 - Formuliere eine kurze Begründung
 - Optional: Winsorisieren
- 👉 Methode wählen, begründen, dokumentieren.

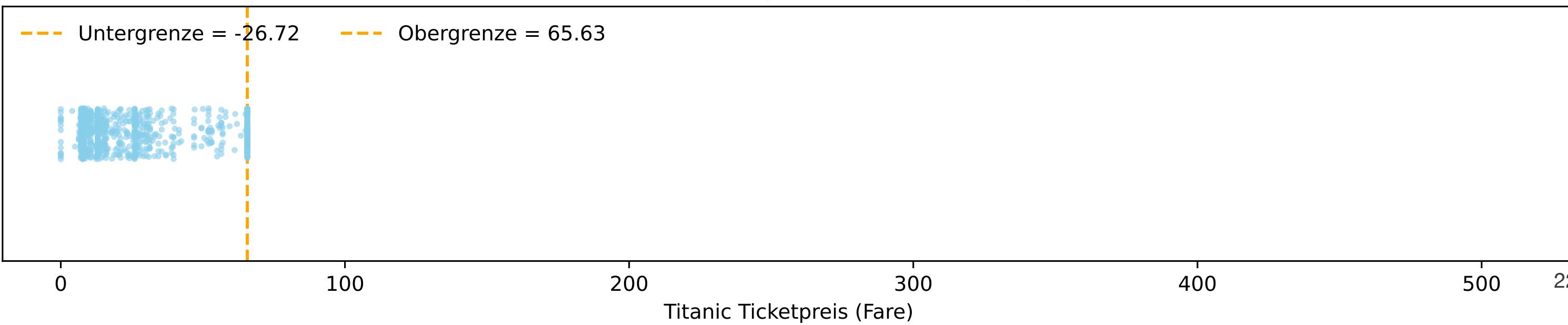
Tukey (1.5·IQR) | Ausreisser: 116 | n (Inlier): 775



Mod. Z-Score ($|M|>3.5$) | Ausreisser: 160 | n (Inlier): 731



Winsorisiert (auf Tukey-Grenzen) | n (innerhalb): 891 | gedeckelt: 116



Checkliste – Ausreisser-Handling

Einheitliche Checkliste für Konsistenz.

- Missing prüfen
- Tukey- und mod. Z-Test anwenden
- Heavy Tails vs. echte Fehler unterscheiden
- Strategie wählen & dokumentieren

Mini-Check: Zwei Punkte, die oft fehlen

👉 Einheitliche Checkliste schützt vor Willkür.



Key Takeaways – Ausreißer erkennen & behandeln

- **Klassischer Z-Score:** basiert auf *Mean* & *SD*, nur bei Normalverteilung sinnvoll
 - **Tukey-Fences:** *IQR*-basiert, robust, Standard im Boxplot
 - **Mod. Z-Score:** *Median* & *MAD*, sehr robust bei Schiefe/Heavy Tails
 - **Einfluss:** *Mean* & *SD* kippen schnell, *Median* & *IQR* bleiben stabil
 - **Strategien:** Markieren, Winsorisieren, Trimen – dokumentieren
 - **Praxis:** Heavy Tails ≠ Fehler → Diagnose vor Eingriff
- 👉 Outlier-Erkennung = Methode wählen + Kontext verstehen + dokumentieren.

Histogramm und KDE

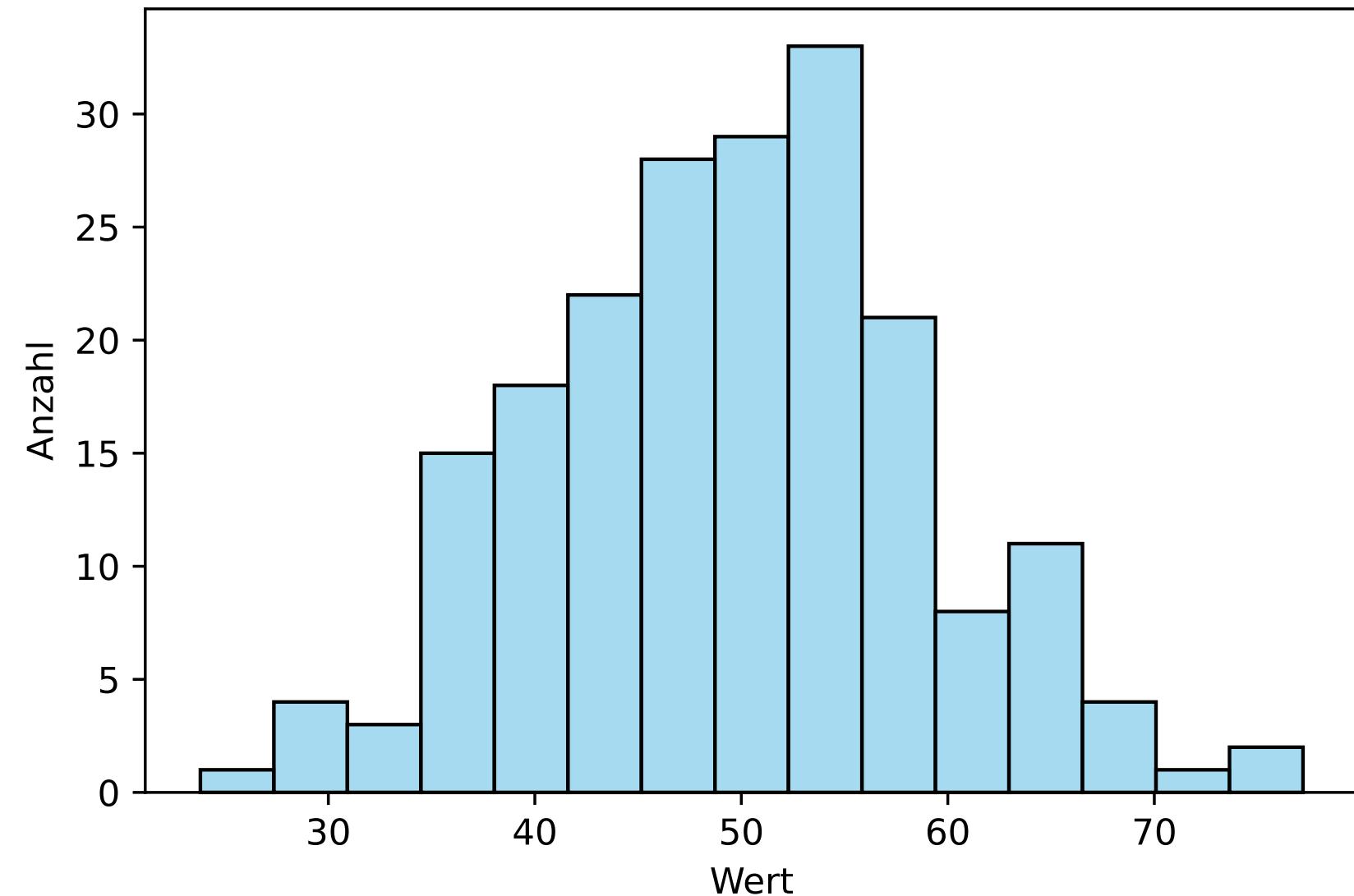
(Kernel Density Estimation –
Kerndichteschätzung)

Histogramm – Grundidee

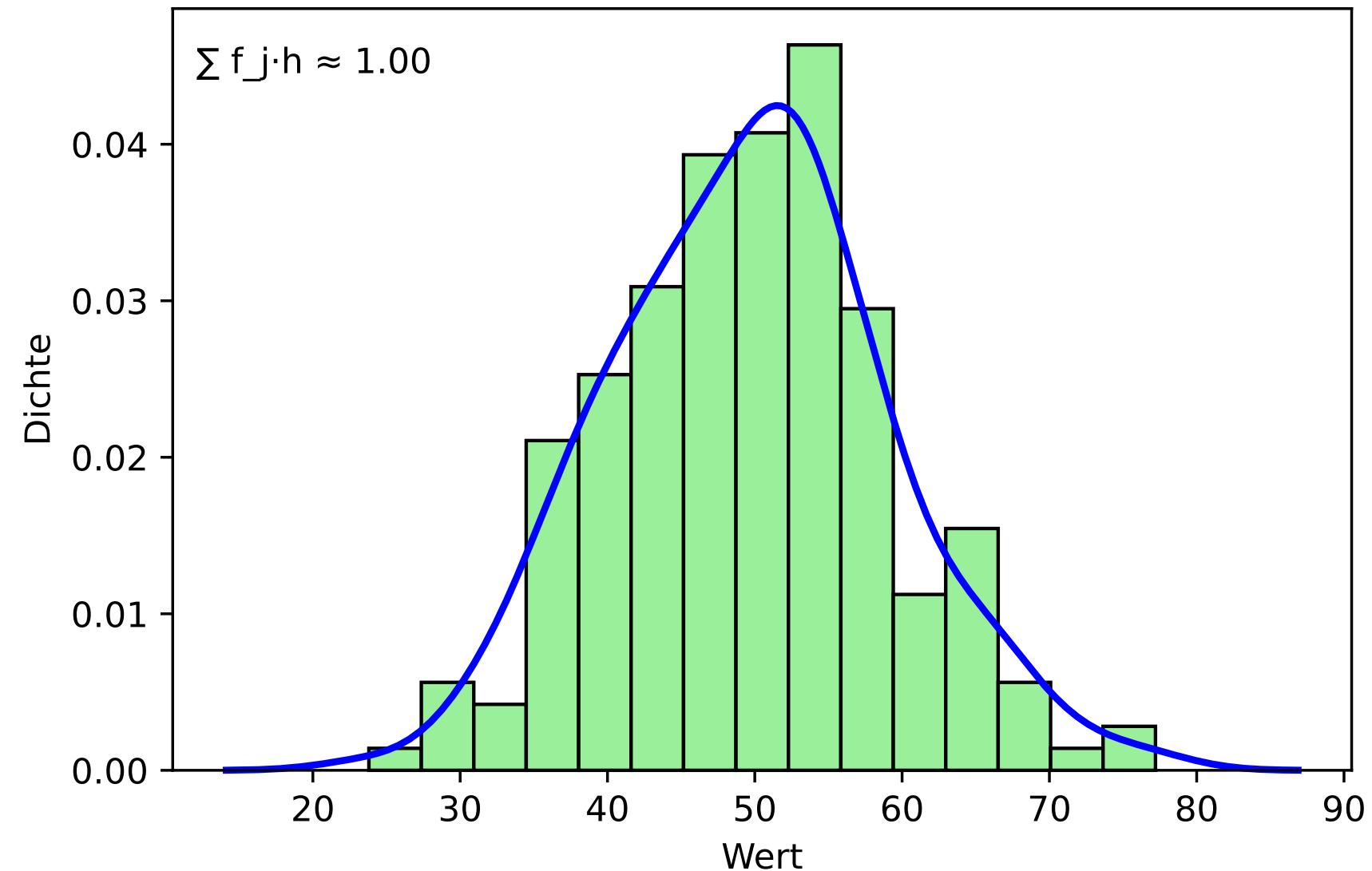
Klassen mit fester Breite zeigen Verteilung.

- Darstellung: Häufigkeit (Counts) oder Dichte
- Dichtehöhe: $f_j = \frac{c_j}{n \cdot h}$
- Fläche der Dichte = 1
- Erkennt Form, Lage und Moden
- **Mini-Check:** Warum ist Fläche 1?
- c_j = Anzahl Werte im Intervall
- n = Gesamtzahl der Werte
- h = Breite des Intervalls

Histogramm mit Counts



Histogramm mit Dichte (Fläche = 1)



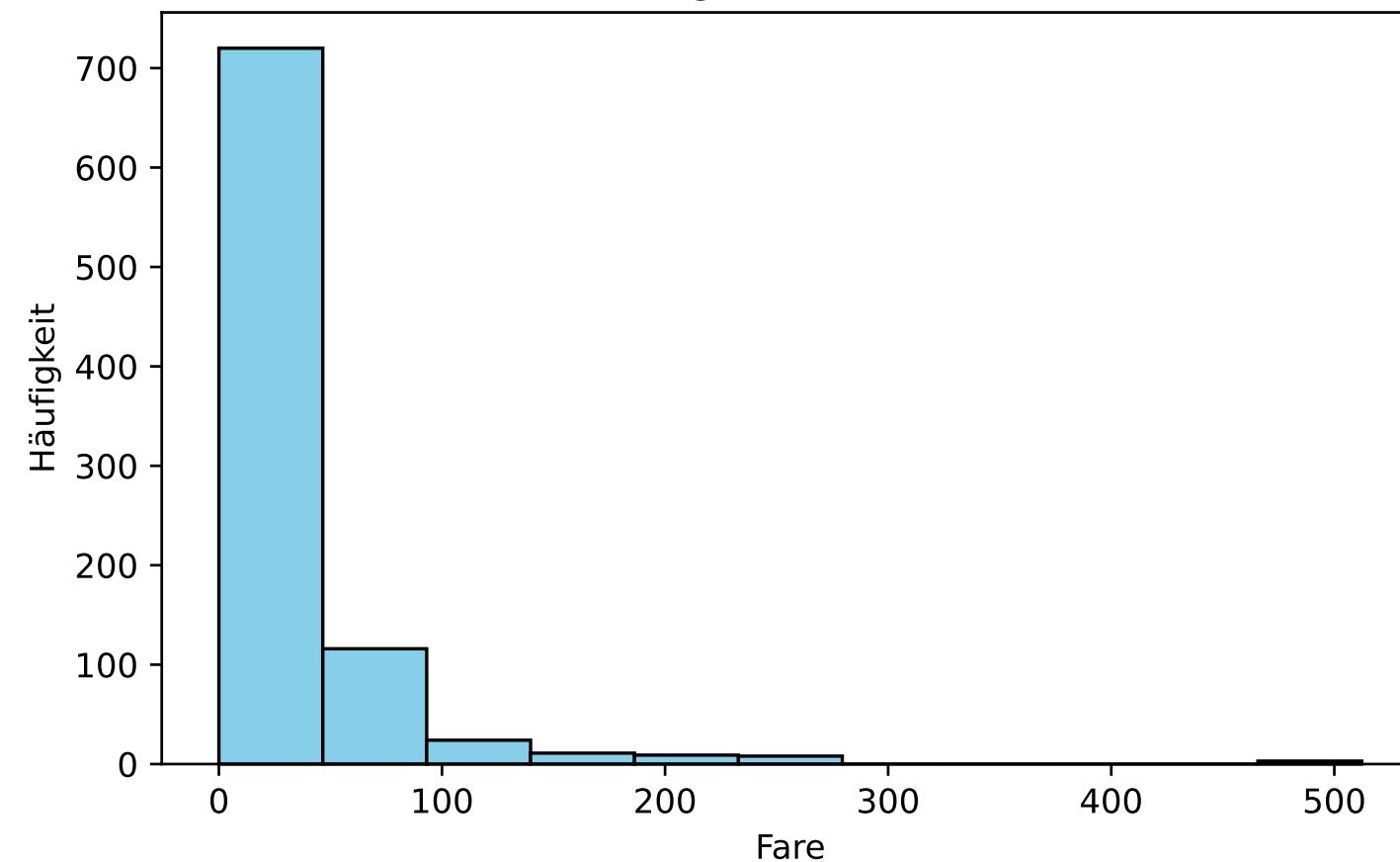
Binwahl – Regeln & Formeln

Bins steuern Aussagekraft.

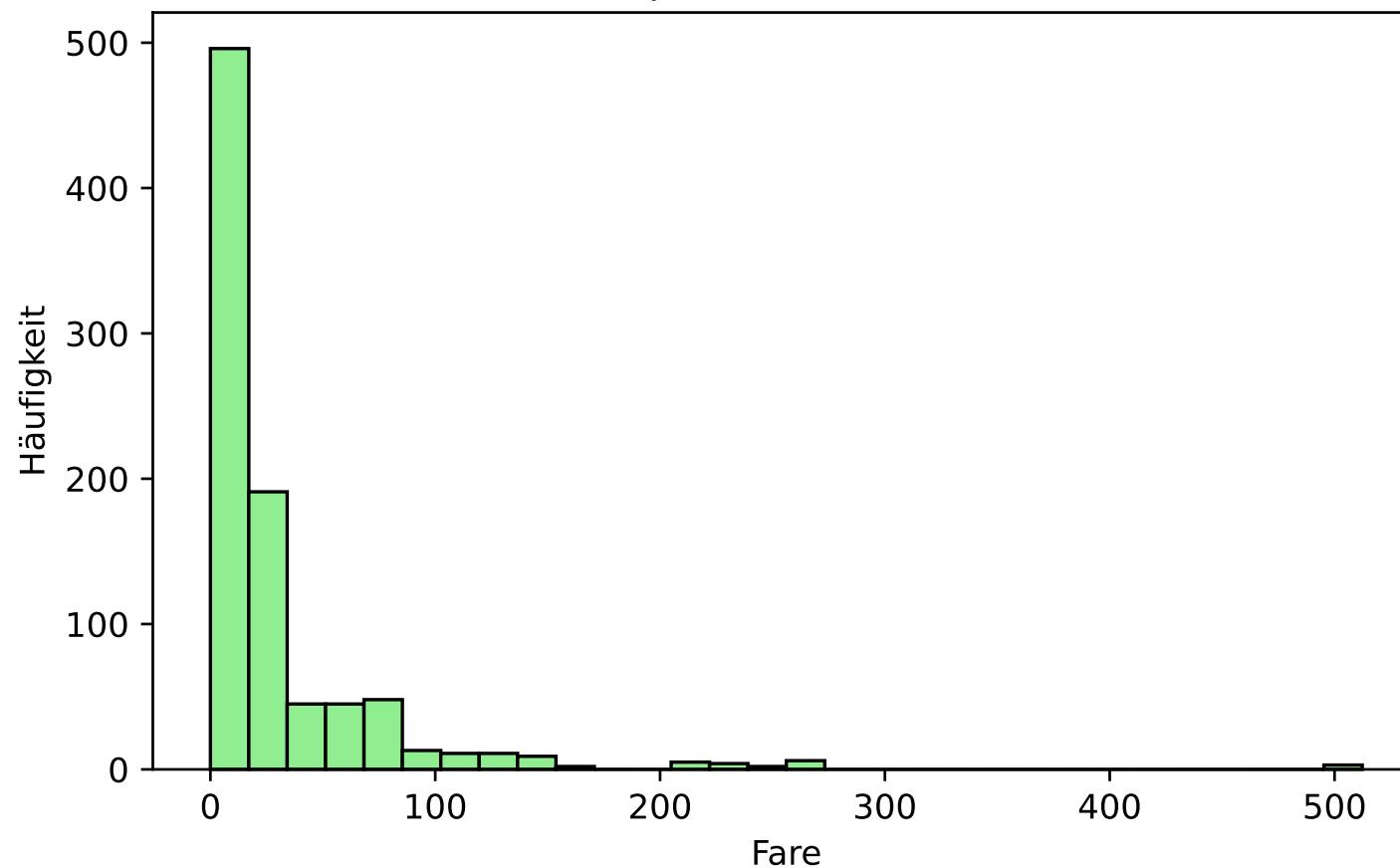
- Faustregeln:
 - Sturges: $B = \lceil \log_2(n) + 1 \rceil$ ← normalverteilungsnah, eher konservativ.
 - \sqrt{n} ← Daumenregel, schnell & simpel.
 - Freedman–Diaconis (FD): $h = 2 \cdot IQR \cdot n^{-1/3}$ ← theoretisch fundiert
- Praxis: FD am robustesten

Mini-Check: Welche Regel bei Schiefe?

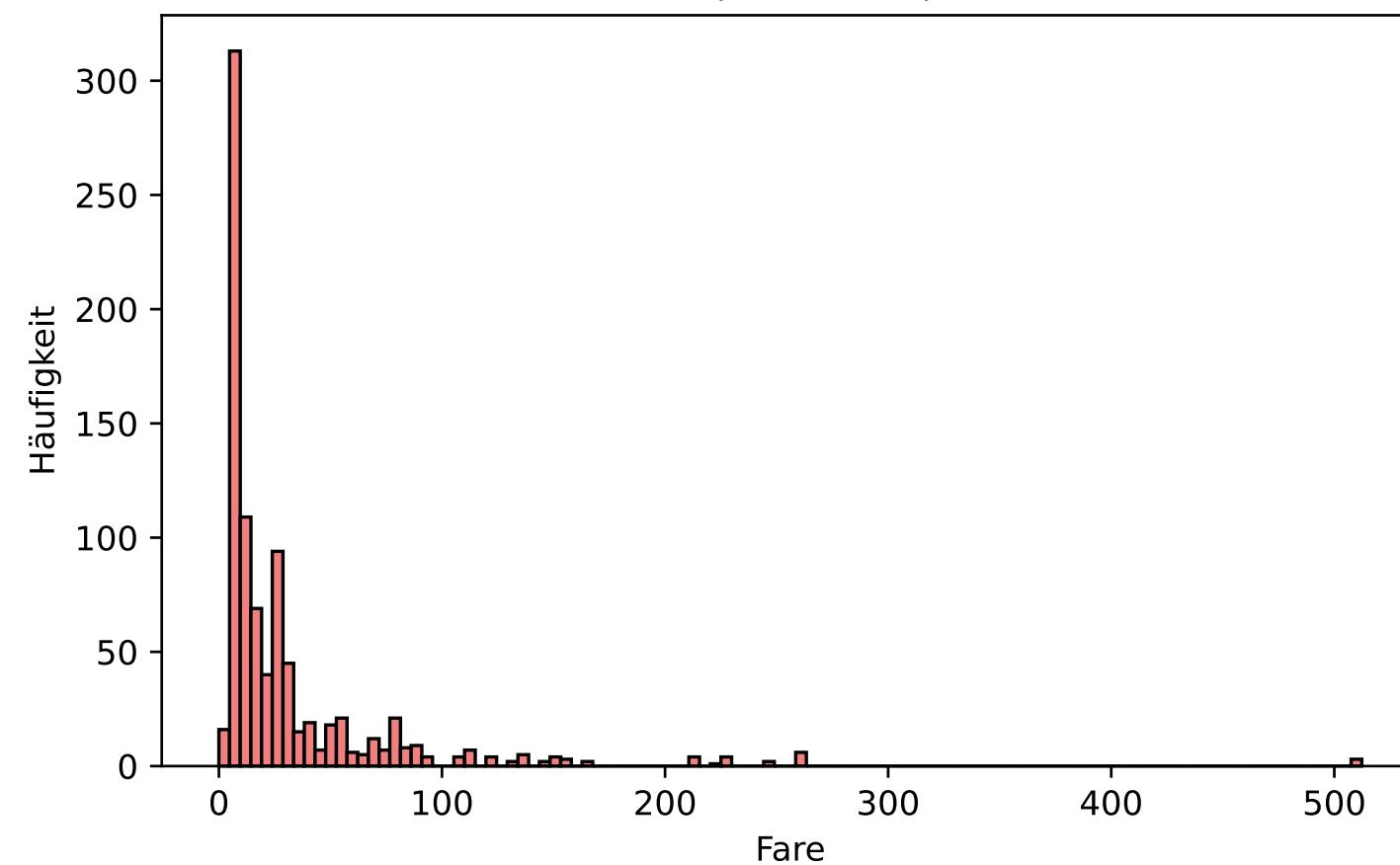
Sturges (Bins=11)



Sqrt(n) (Bins=30)



FD (Bins=107)



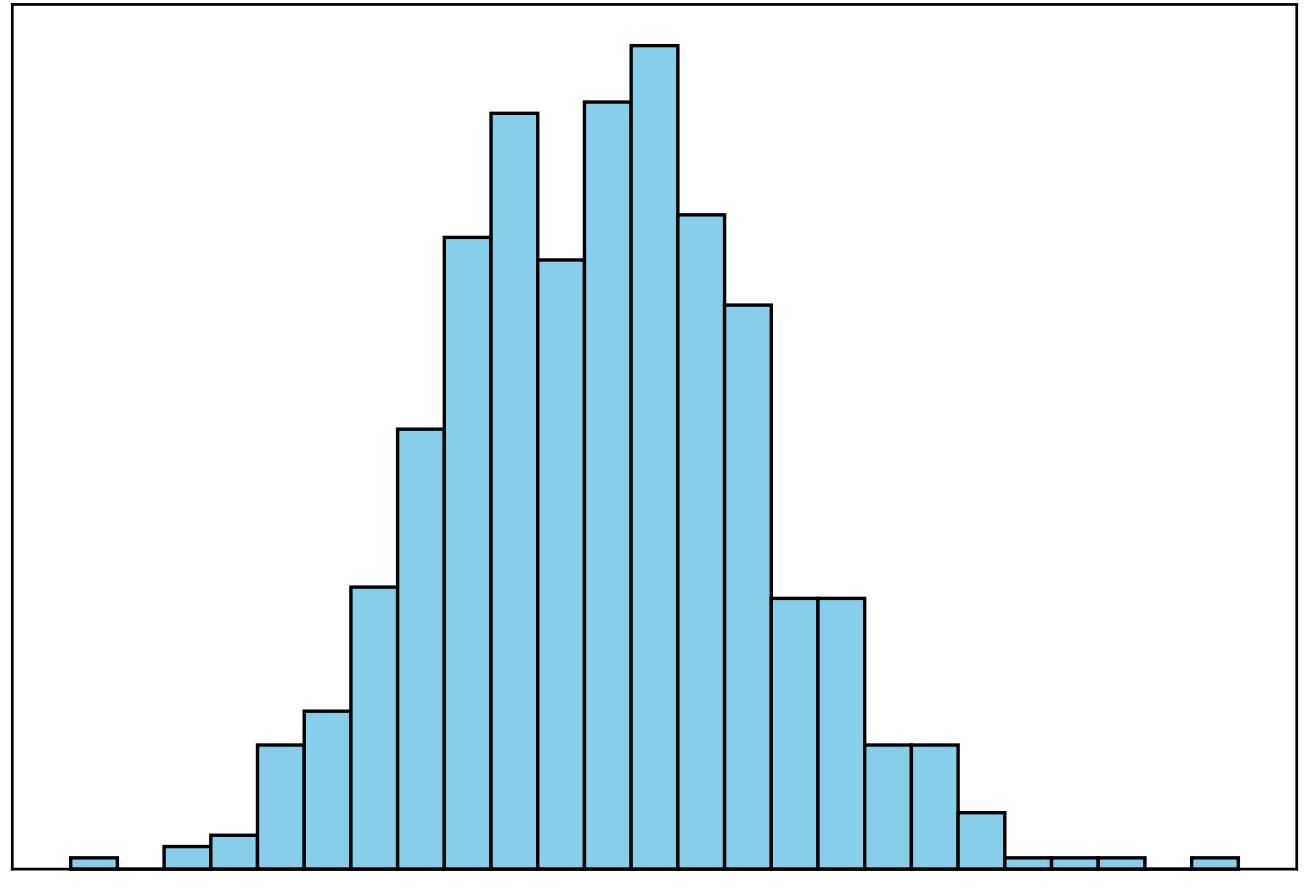
Histogramme lesen

Interpretation geht über Zählen hinaus.

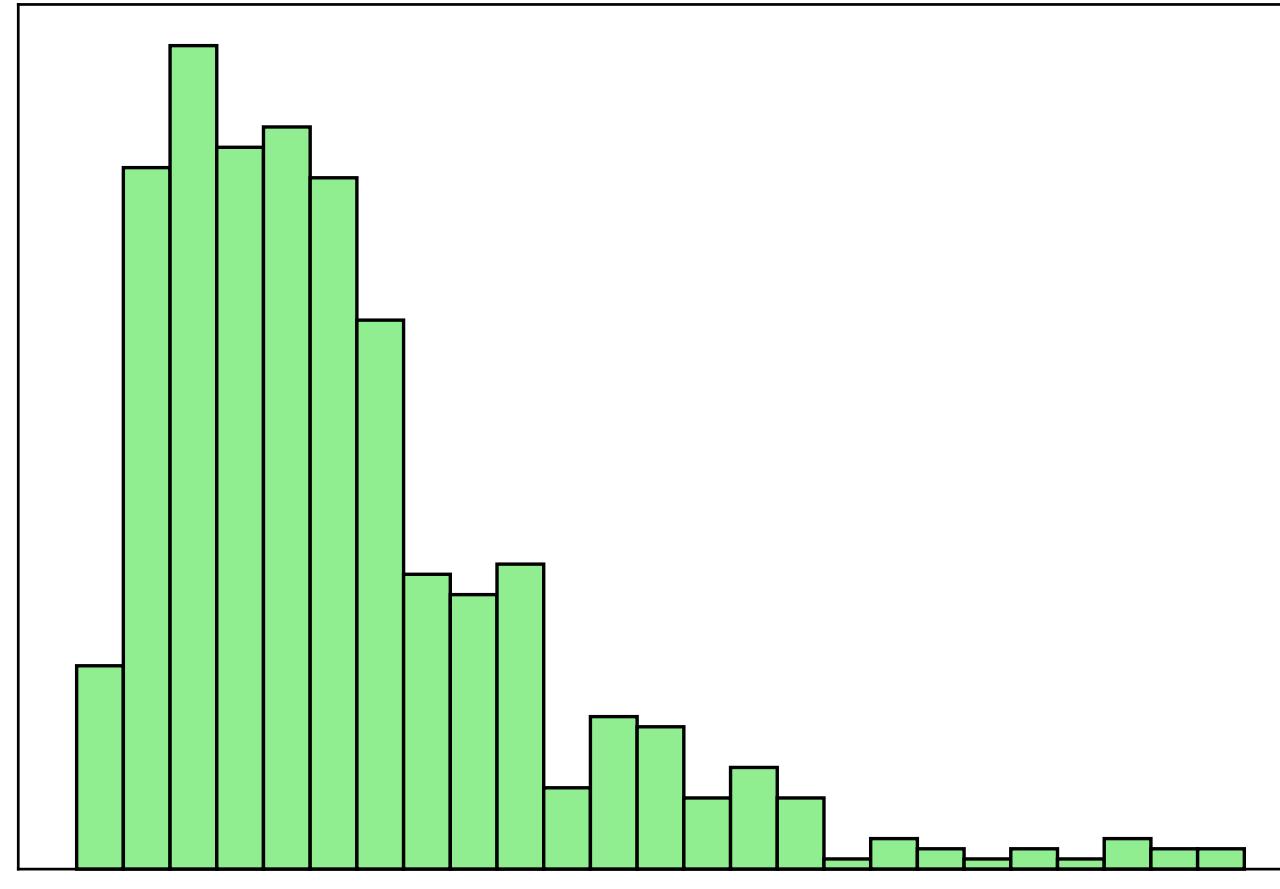
- Schiefe erkennen
- Moden zählen
- Lücken deuten
- Tails prüfen

Mini-Check: Was bedeutet Lücke in der Mitte?

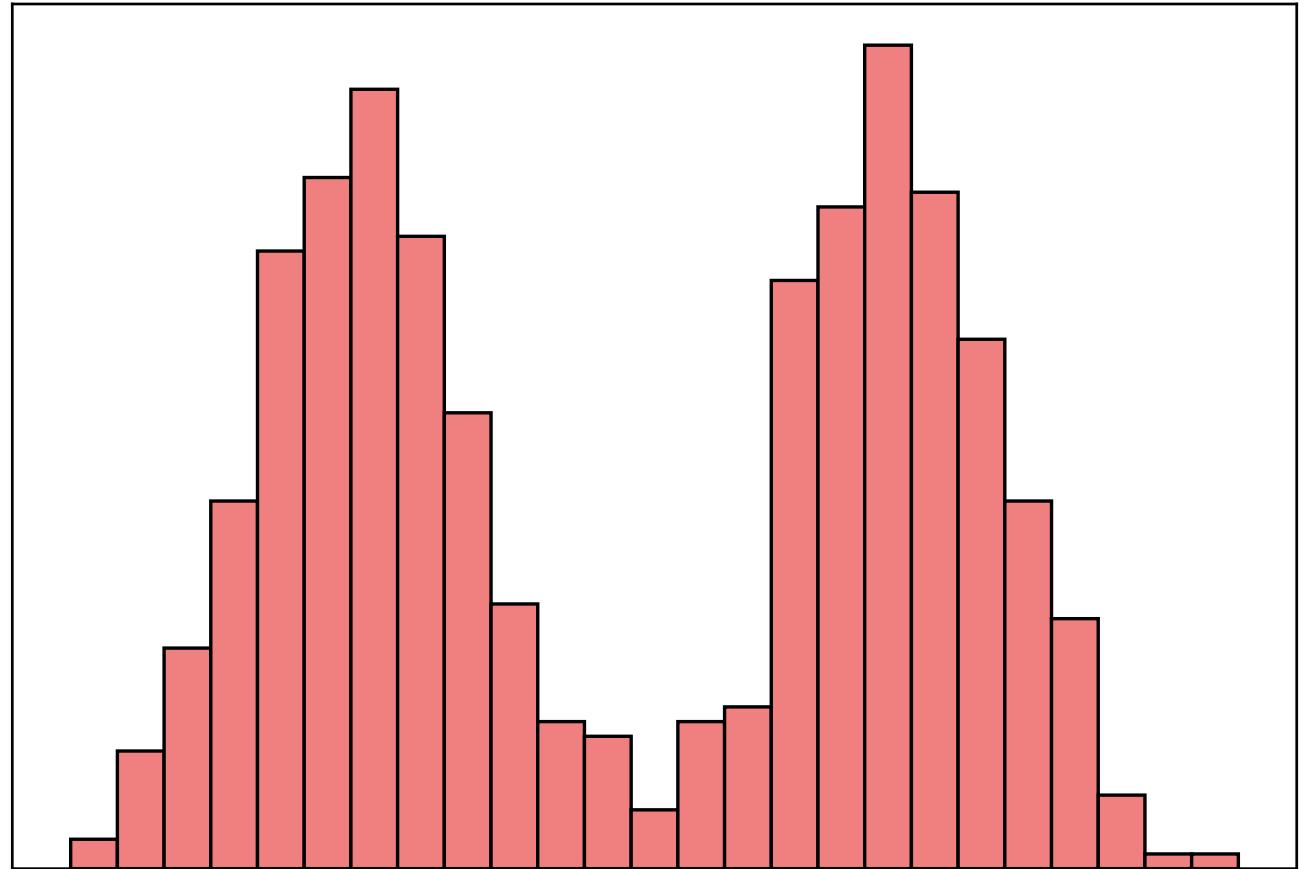
Symmetrisch



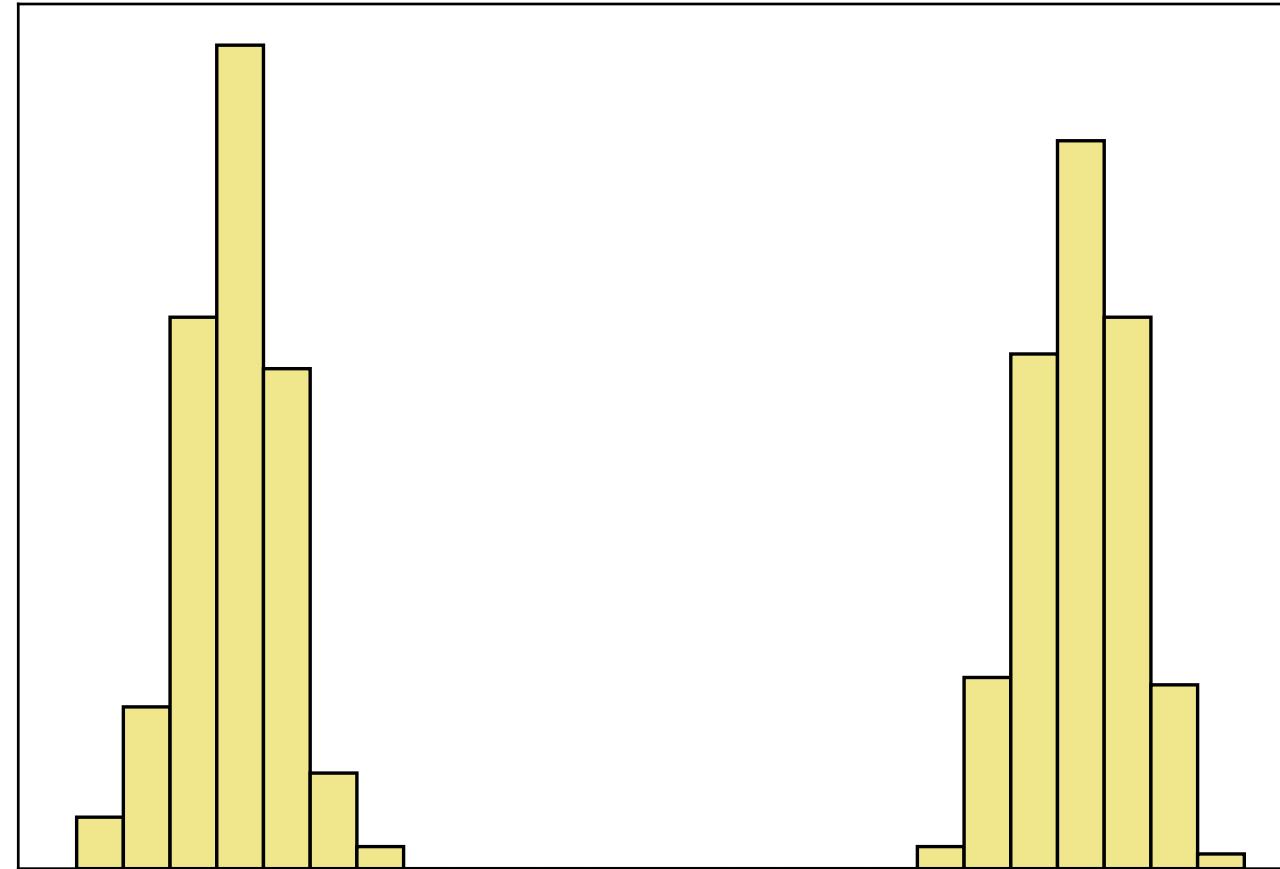
Rechts-schief



Bimodal



Mit Lücke



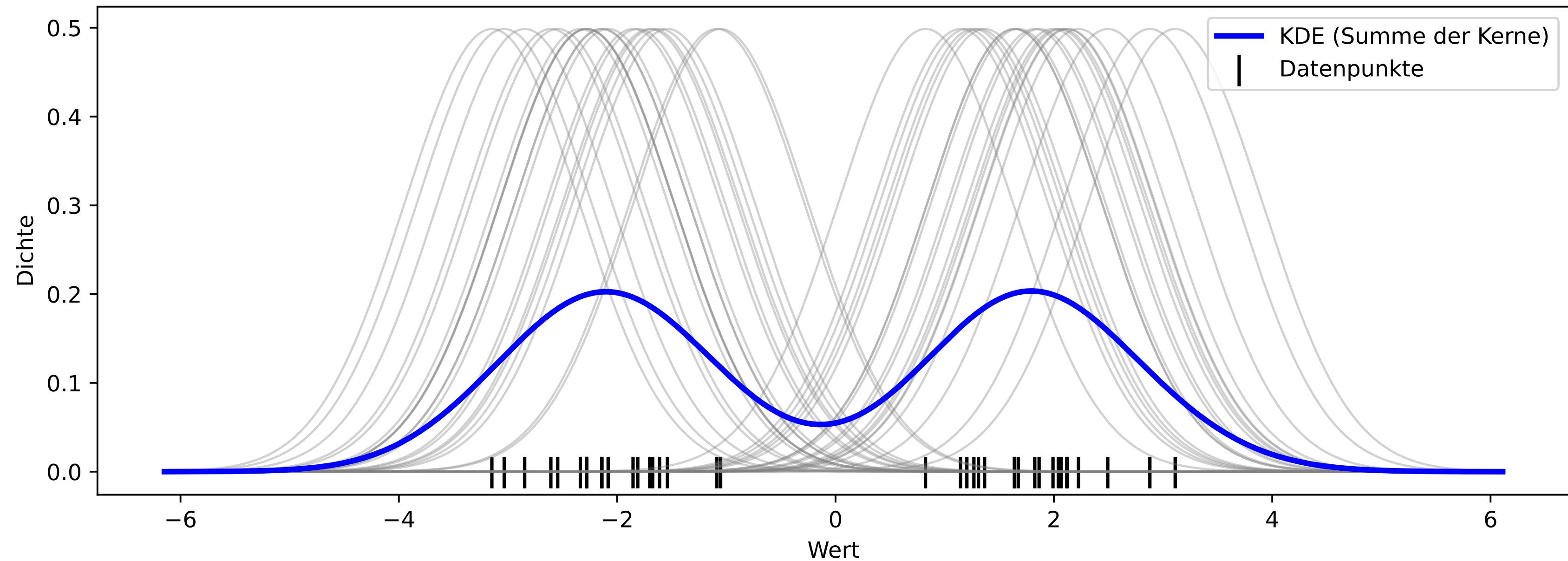
KDE (Kerndichteschätzung) – Idee der Glättung

Bandbreite ersetzt Bins

- Schätzung: $\hat{f}_h(x) = \frac{1}{nh} \sum K\left(\frac{x-x_i}{h}\right)$
- Üblich: Gauß-Kern
- Parameter h = Bandbreite steuert Glättung
- Bin-frei, aber parameterabhängig
- n : Anzahl Daten
- h : Bandbreite (Glättung)
- x_i : i-ter Datenpunkt
- K : Kern (z. B. Gauß-Glocke)
- $\frac{x-x_i}{h}$: Distanz zu x_i in h -Einheiten

Mini-Check: Warum nicht parameterfrei?

KDE als Summe von Gauss-Kernen (Bandbreite $h=0.8$)

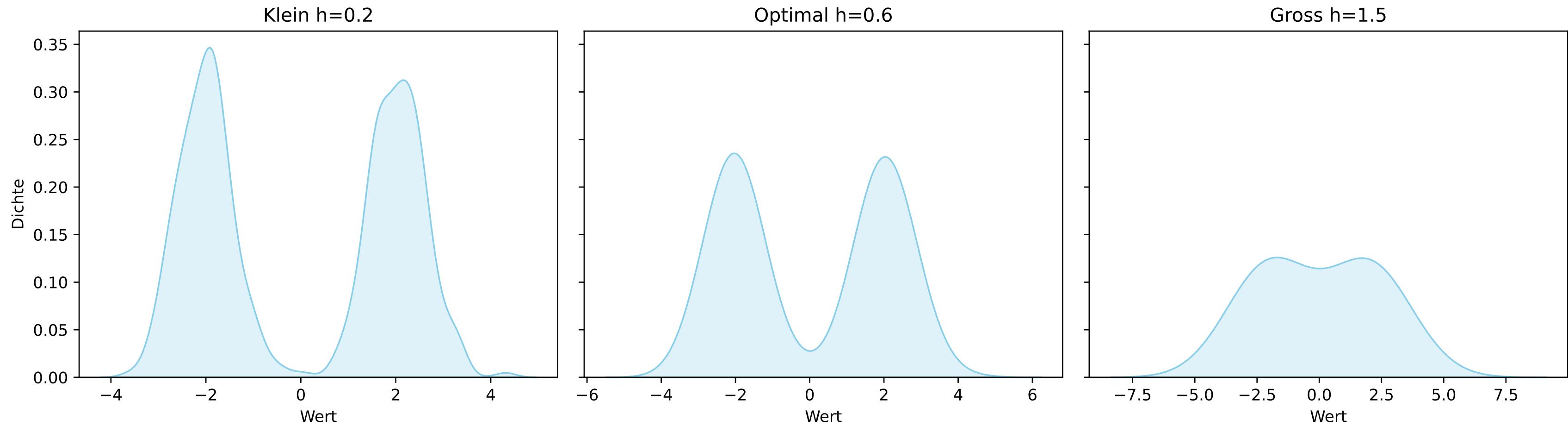


Bandbreite wählen

Unter- vs. Überglättung unterscheiden.

- Kleine $h \rightarrow$ viele Details, evtl. Rauschen
- Grosse $h \rightarrow$ geglättet, Details verschwinden
- Heuristiken:
 - **Scott:** $h = \sigma n^{-1/5} \sqrt{n}$ \leftarrow normalverteilten Daten
 - **Silverman:** $h = 0.9 \cdot \min(\sigma, IQR/1.34) n^{-1/5}$ robuster bei schießen Daten

Mini-Check: Wann Silverman statt Scott?



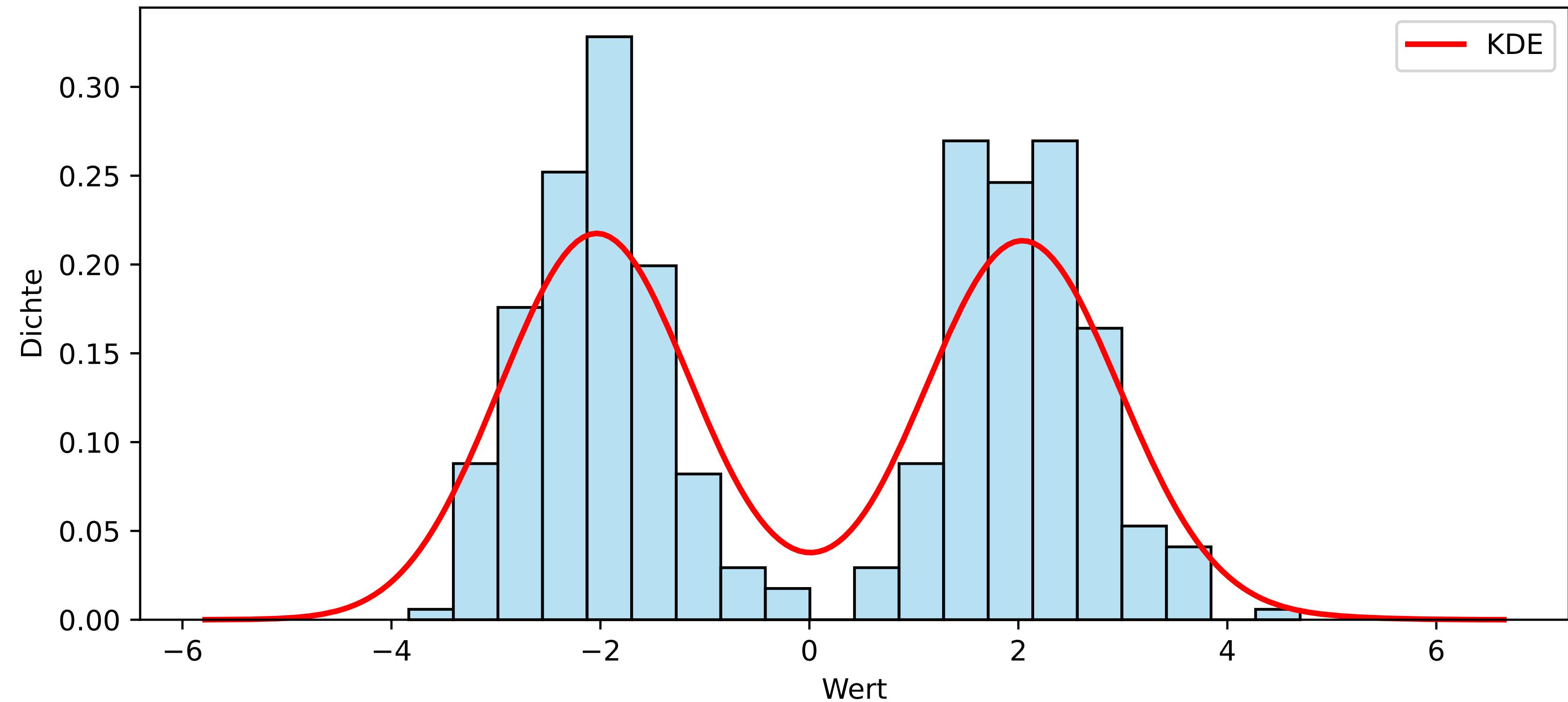
Histogramm oder KDE?

Kombination ist oft ideal

Methode	Eigenschaften
Histogramm	Intuitiv, zeigt Häufigkeiten; Stabil bei kleinem n
KDE	Glatte Form, Moden erkennbar; Parameterabhängig h

Beste Praxis: Overlay von Histogramm + KDE

Histogramm + KDE Overlay



Python Histogramm und KDE

Saubere Defaults genügen.

```
plt.hist(x, bins="fd", density=True, edgecolor="black", alpha=0.4)  
sns.kdeplot(x=x, bw_adjust=1.0)
```

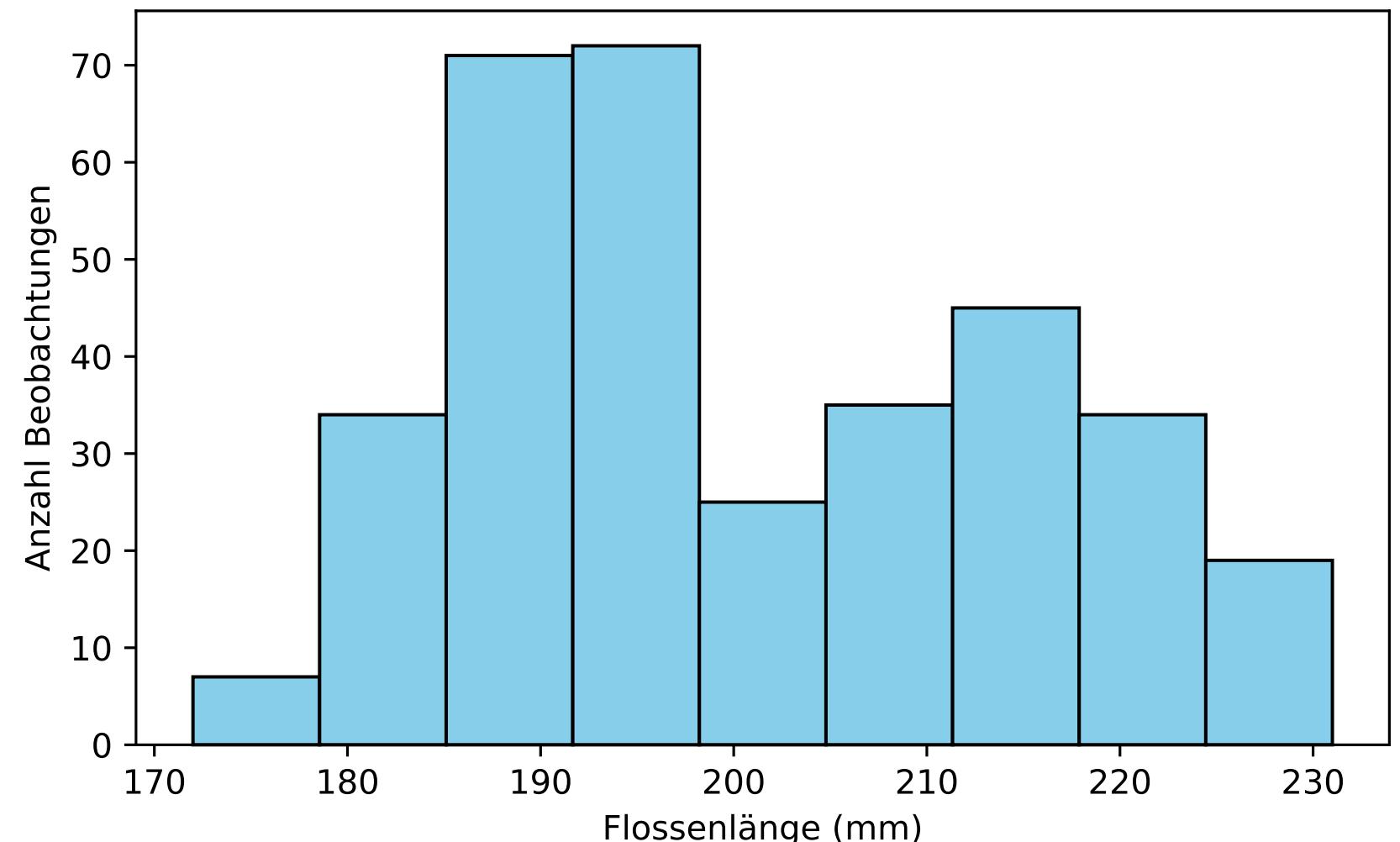
- `bw_adjust` variieren → Bandbreite anpassen

Mini-Check: Warum `density=True`?

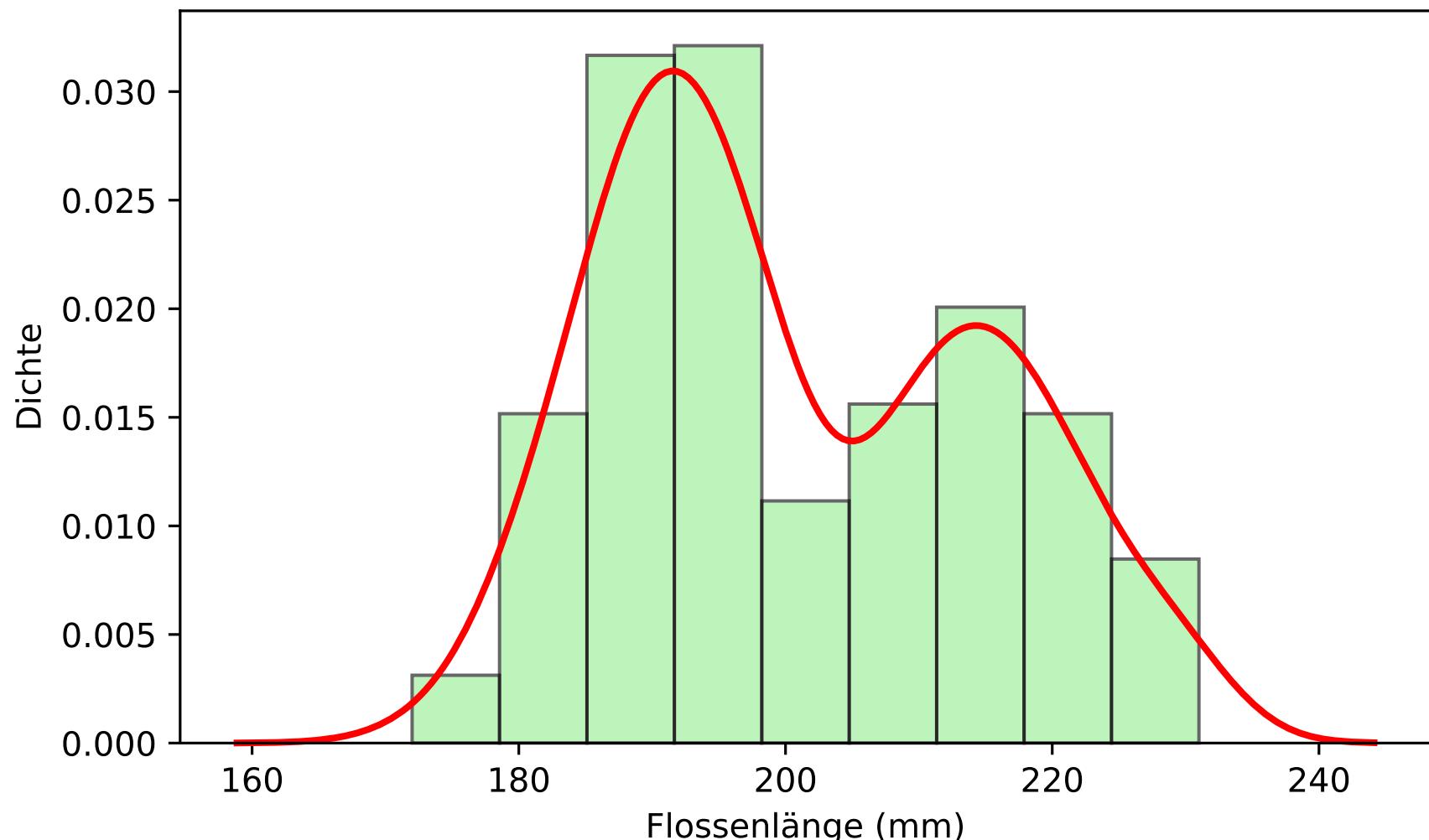
bw_adjust in der KDE

- **Was ist es?** Skalar, der die automatisch gewählte Bandbreite h_0 skaliert: $h = \text{bw_adjust} \cdot h_0$
- **Default & Heuristik:** h_0 aus Scott/Silverman (grob $h_0 \propto n^{-1/5}$). **bw_adjust = 1.0** ist Standard.
- **Wirkung:** < 1 = feiner, mehr Detail, Risiko Rauschen. > 1 = glatter, weniger Detail, Risiko Strukturverlust.
- **Zielbild:** Moden sichtbar, keine Zacken. Entscheidung kurz dokumentieren.
- **Vergleich:** Immer gegen Dichte-Histogramm prüfen (Fläche = 1).
- **Mehrdimensional:** Skaliert Bandbreitenmatrix gleichmäßig; zu gutes Setting kann Cluster verschmelzen.
- **Praxiswerte:** 0.5, 1.0, 2.0 testen. Bei kleinem n eher konservativ (≥ 1.0).
- **Merke:** Form ändert sich, die Fläche bleibt 1.

Histogramm mit Counts



Histogramm mit Dichte (Fläche = 1) + KDE



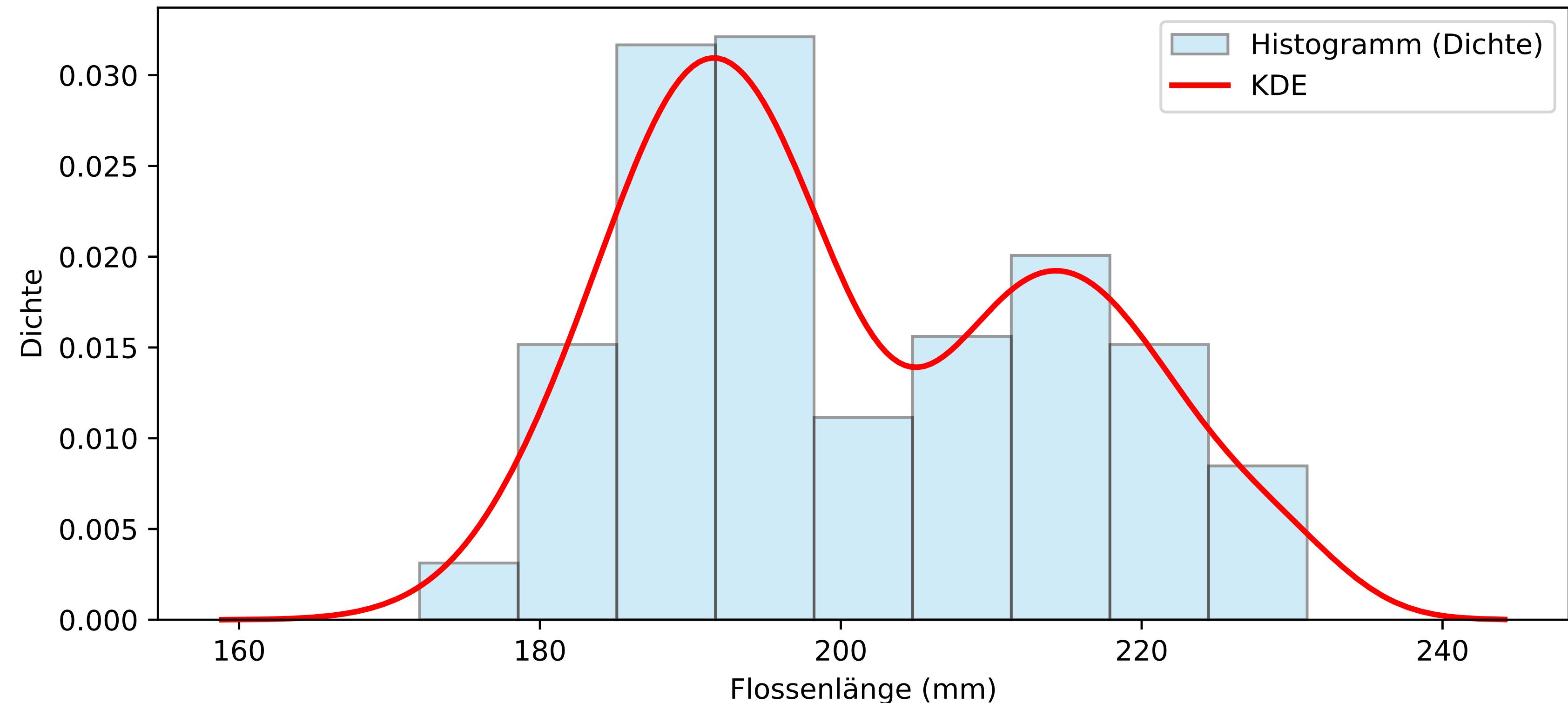
Beispiel Penguins flipper_length

Overlay zeigt Moden und Lage.

```
df = sns.load_dataset("penguins").dropna(subset=["flipper_length_mm"])
x = df["flipper_length_mm"].to_numpy()
plt.hist(x, bins="fd", density=True, alpha=0.4); sns.kdeplot(x=x)
```

Mini-Check: Erkennst du Multimodalität?

Penguins - Flossenlänge (Overlay Histogramm + KDE)



Häufige Fehler: Histogramm & KDE

Setup-Fehler vermeiden.

- Counts und Dichte verwechseln
- Zu wenige Bins → Muster verdeckt
- Falsche Achsenbeschriftung (Counts oder Dichte), Log ohne Hinweis...
- Falsche Bandbreite

Mini-Check: Wie zeigst du Überglättung schnell?

Mini-Aufgabe: Binzahl & Bandbreite

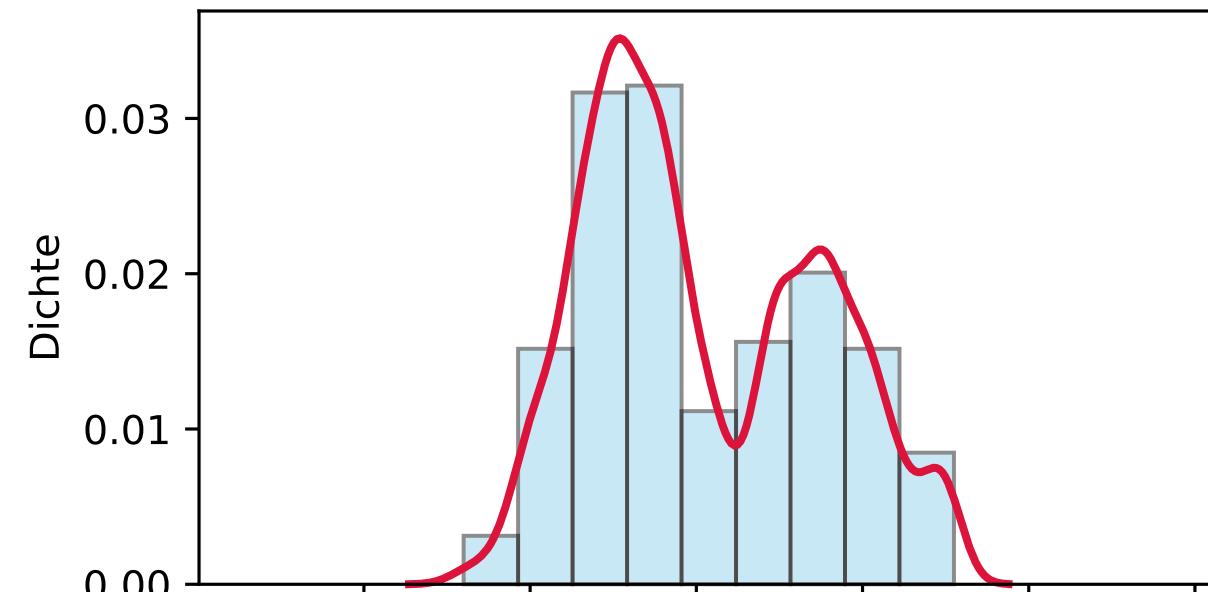
Regel anwenden und verteidigen.

- Wähle Bins mit FD-Regel vs. \sqrt{n}
- Teste bw_adjust in {0.5, 1.0, 2.0}
- Formuliere Entscheidung in 1 Satz
- Kurz begründen

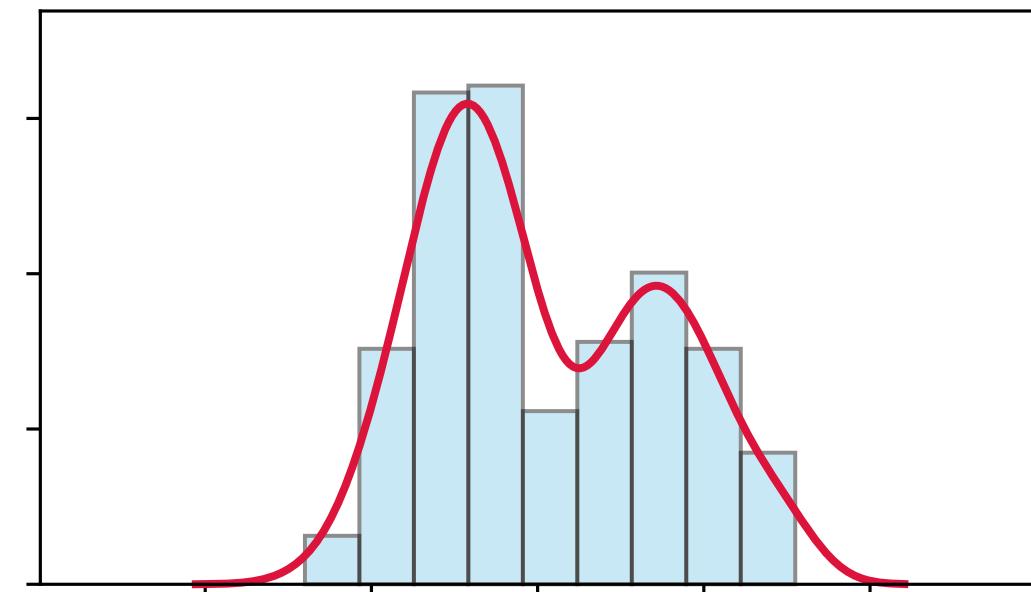
Mini-Check: Warum ändert h die Schlussfolgerung

Penguins – Einfluss von Binzahl (FD vs. \sqrt{n}) und Bandbreite (bw_adjust)

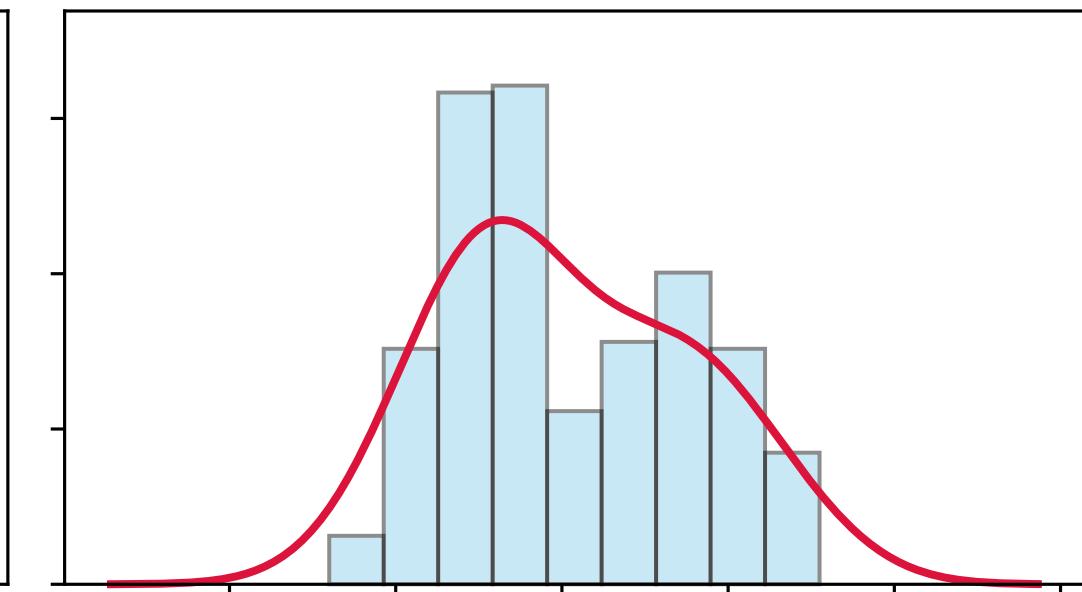
FD-Bins, bw=0.5



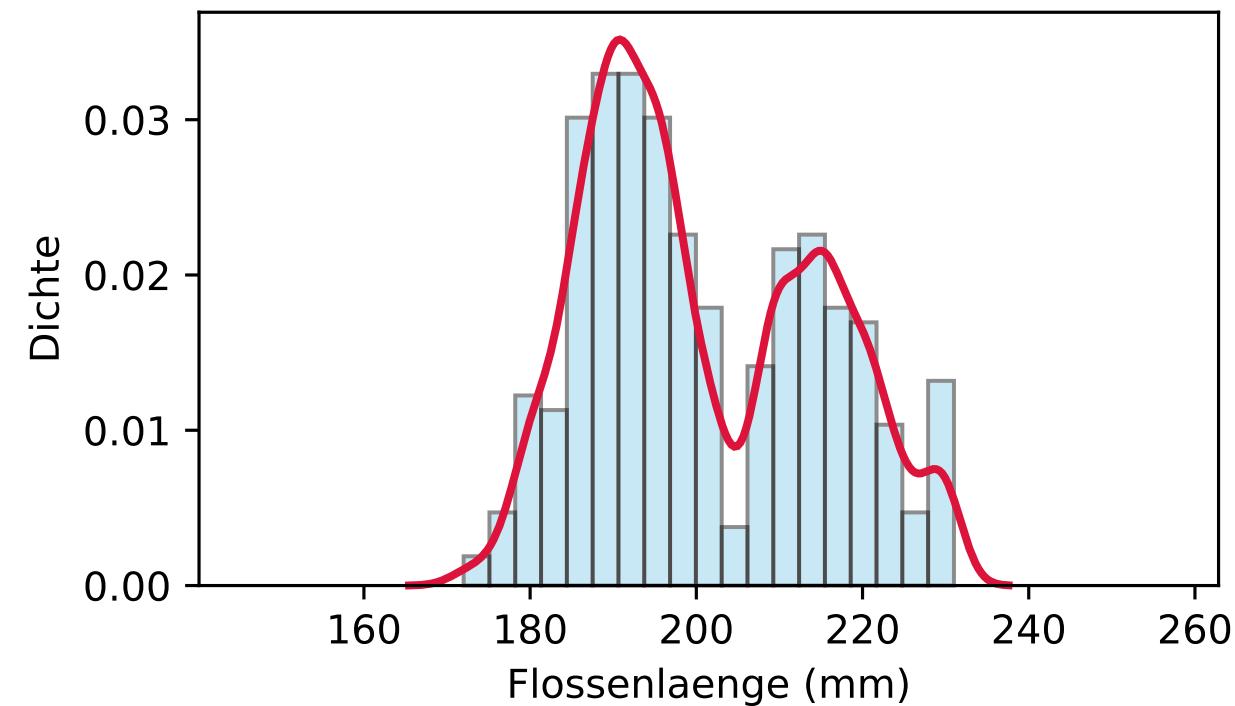
FD-Bins, bw=1.0



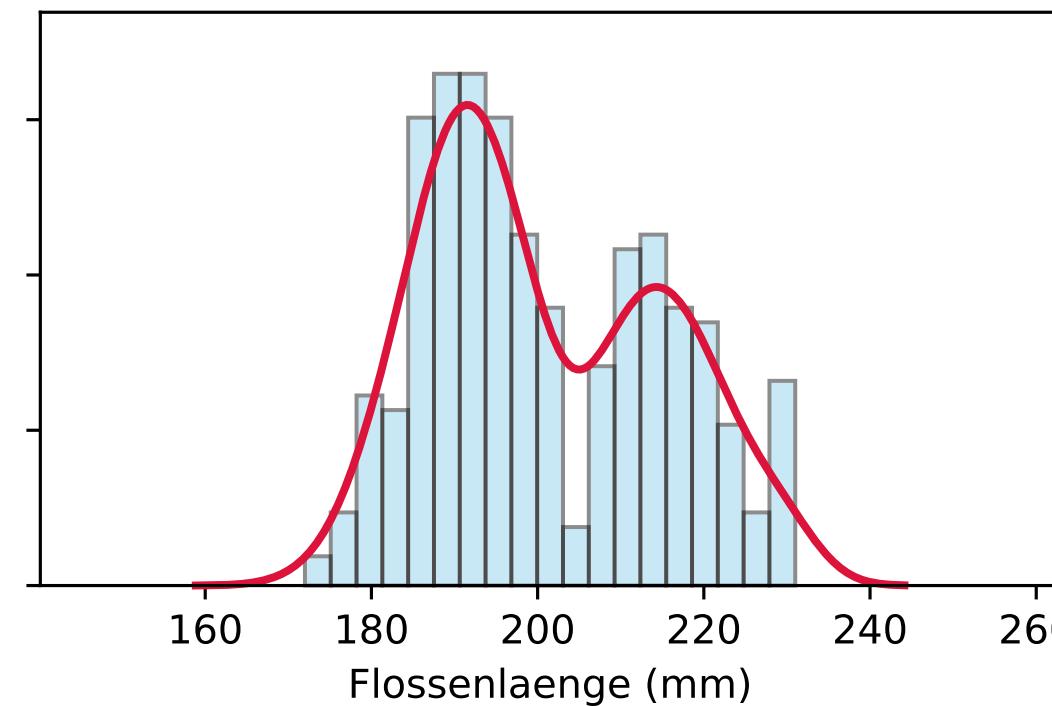
FD-Bins, bw=2.0



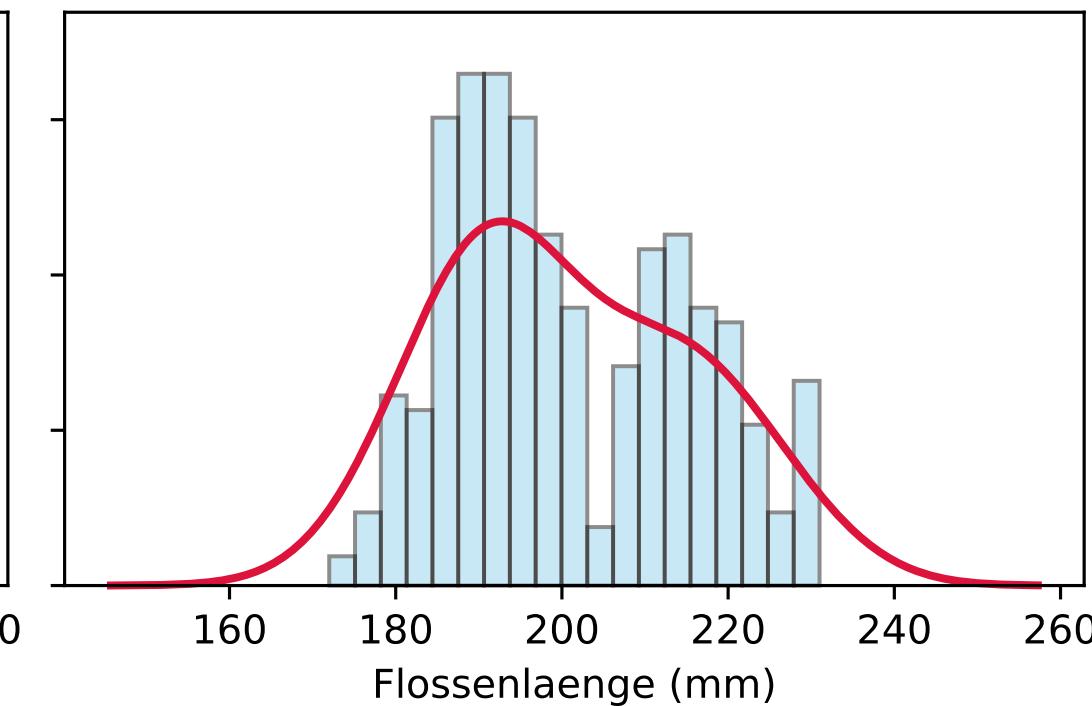
\sqrt{n} -Bins (19), bw=0.5



\sqrt{n} -Bins (19), bw=1.0



\sqrt{n} -Bins (19), bw=2.0



Reporting

Schreiben Sie nach der Analyse **einen Satz** zur Entscheidung:

- Was zeigt die Struktur am besten?
- Ohne zu rauschen, und **warum**?

Musterbeispiel:

«FD-Bins plus KDE mit bw_adjust=1.0 zeigen zwei stabile Gipfel und einen klaren Schwerpunkt um 195 mm. Die Aussage bleibt robust gegenüber bw_adjust 0.5 und 2.0.»

👉 Kurz, prägnant, reproduzierbar.

Feld	Eintrag
Regel	Tukey $1.5 \text{ } IQR + \text{mod.}Z \text{ } 3.5$
Schwelle & Grund	Robuste Erkennung, da Schiefe vermutet
Datum & Version	Datenstand 2025-09-20, Codeversion v1.3
Variable	flipper_length_mm
Ergebnis	5 Kandidaten, 3 real, 2 Messfehler
Entscheid	Markieren, Bericht mit und ohne

Key Takeaways – Histogramm & KDE

- **Histogramm**: zeigt Form, Lage & Moden; Fläche = 1 bei Dichte
 - **Binwahl**: Regeln (Sturges, \sqrt{n} , FD); FD ist robustester Default
 - **Interpretation**: Schiefe, Moden, Lücken & Tails beachten
 - **KDE**: glatte Alternative zum Histogramm, abhängig von Bandbreite h
 - **Bandbreite**: zu klein = Rauschen, zu gross = Details verschwinden
 - **Praxis**: Overlay Histogramm + KDE liefert das beste Bild
- 👉 Robustheitsmasse geben realistischere Bilder als nur der Mittelwert.

Box und violin

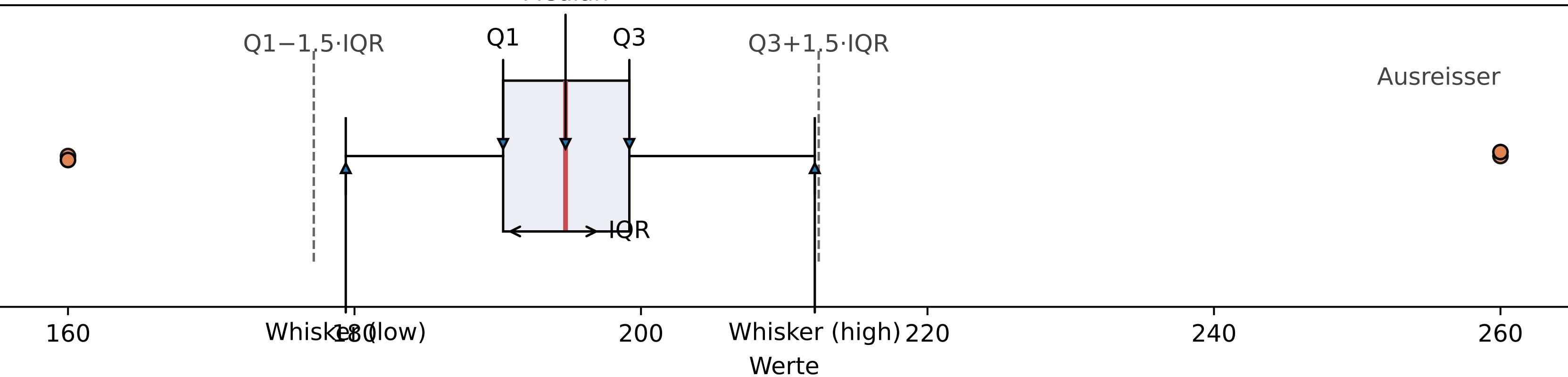
Boxplot Anatomie

Robuste Lage und Streuung auf einen Blick.

- Median, Box Q_1 bis Q_3 , IQR
- Whisker bis $Q_1 - 1.5 \cdot IQR$ und $Q_3 + 1.5 \cdot IQR$
- Punkte ausserhalb als Kandidaten
- Robust gegenüber wenigen Extremen

Mini-Check: Was ändert sich bei grösserem IQR?

Boxplot Anatomie - Q1, Median, Q3, IQR, Whisker, Fences



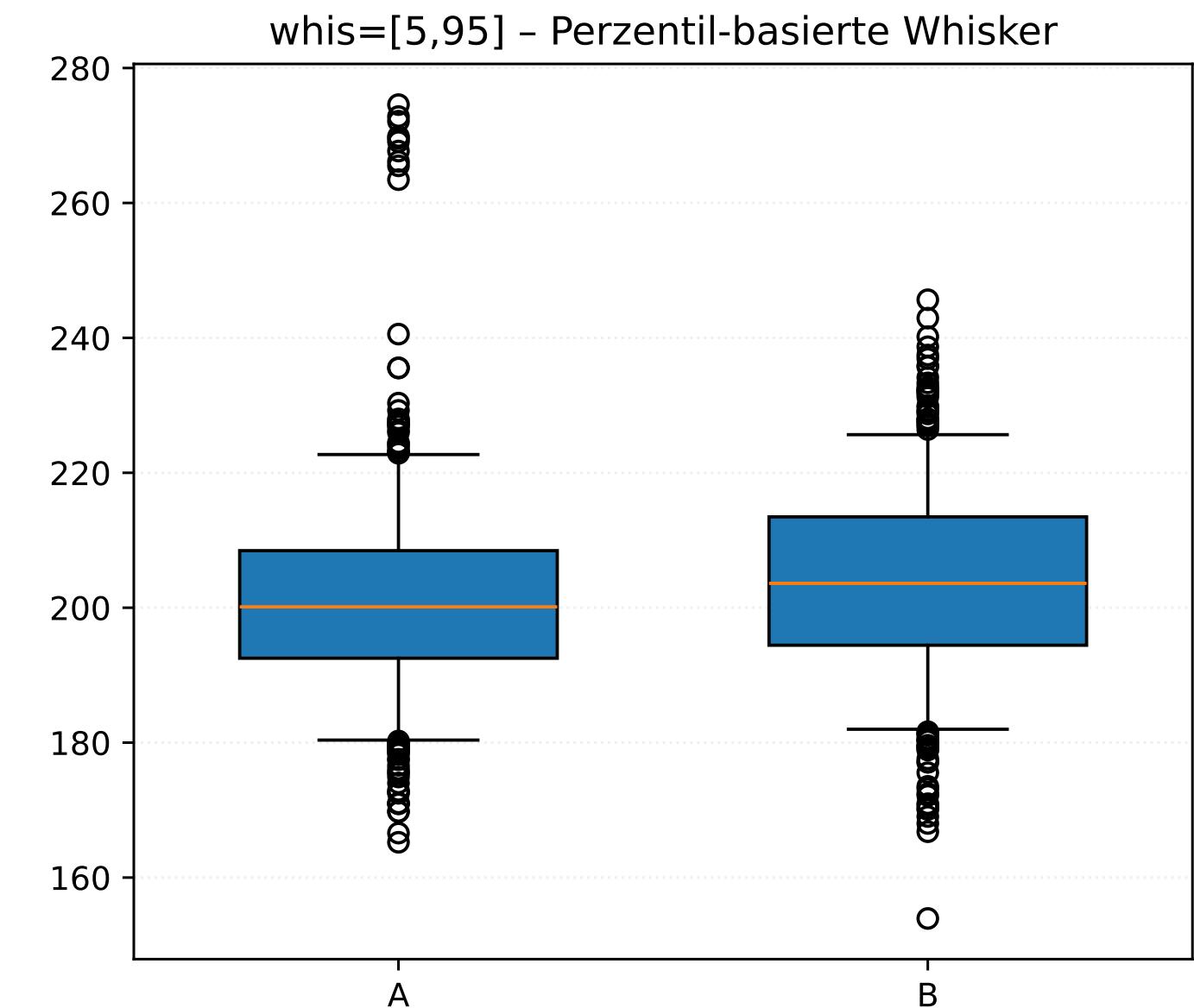
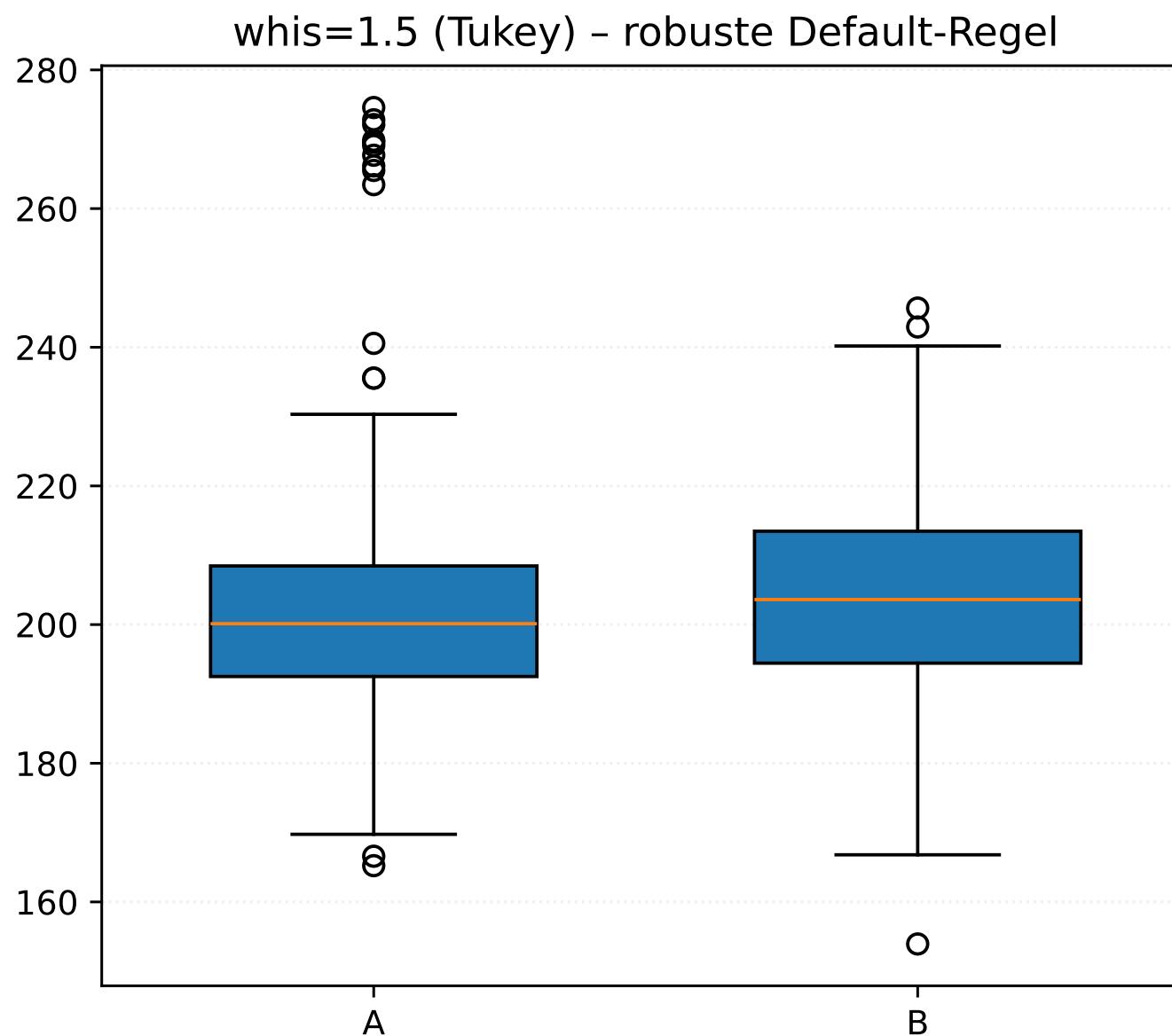
Boxplot Varianten und Parameter

Parameter steuern Aussage.

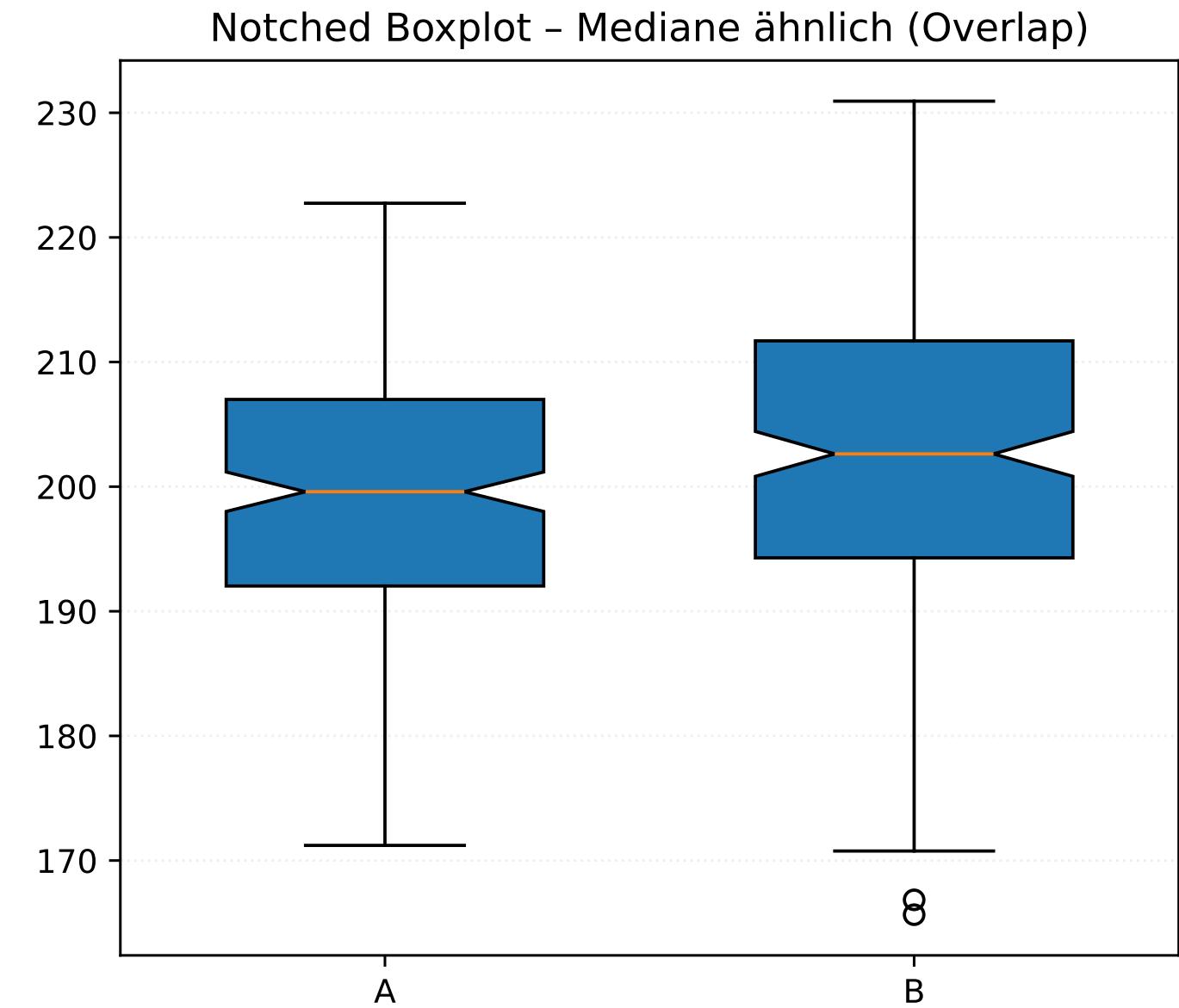
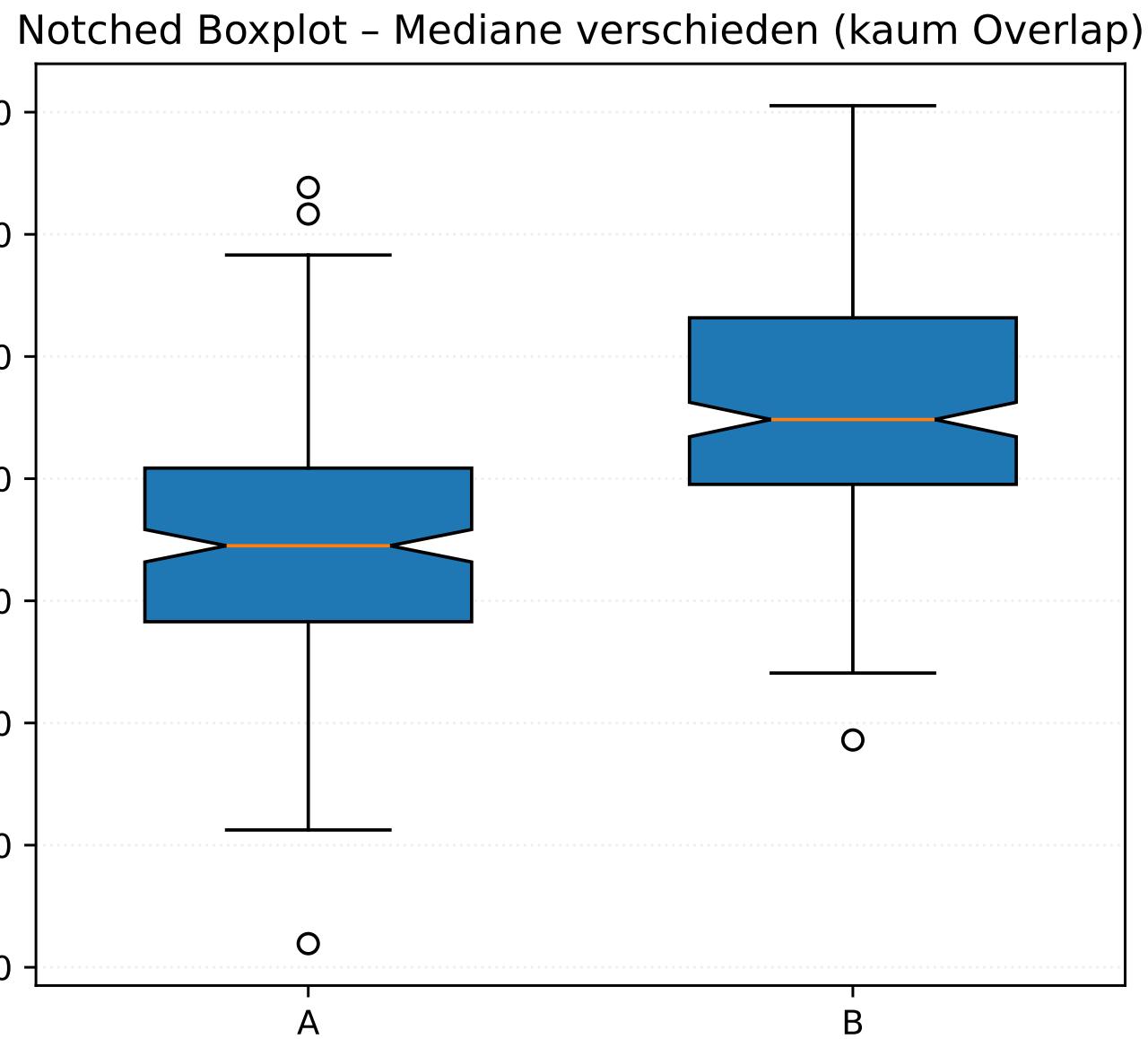
- `whis=1.5` → klassische Tukey-Regel
- Alternative: Perzentile z. B. `whis=[5, 95]`
- **Notches**: zeigen Konfidenzintervall des Medians
- **Log-Skala**: nützlich bei starker Schiefe
- Mehrere Gruppen nebeneinander vergleichbar

Mini-Check: Wann `whis=[5, 95]` wählen?

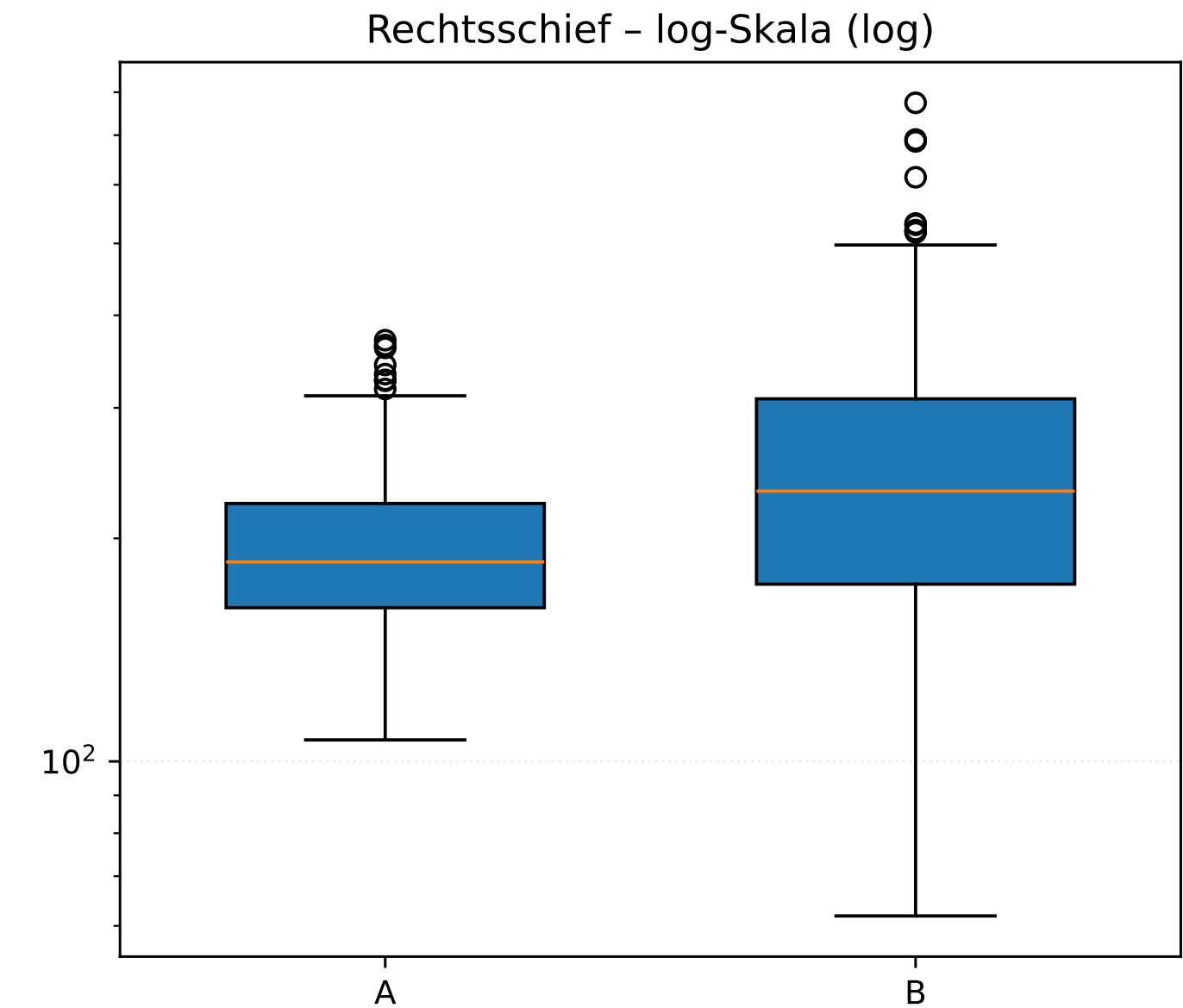
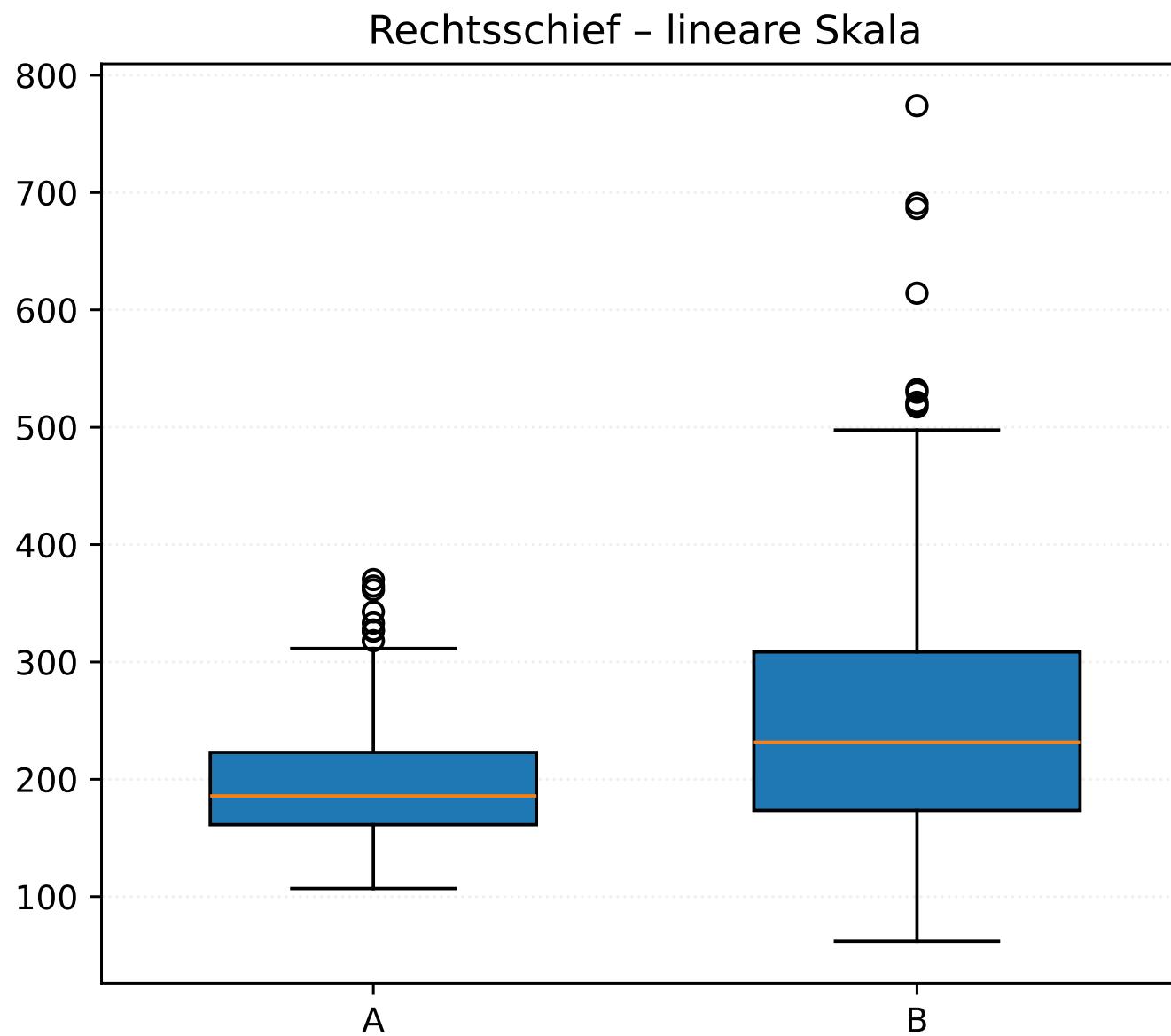
Whisker?



Notch?



Linear oder Log?



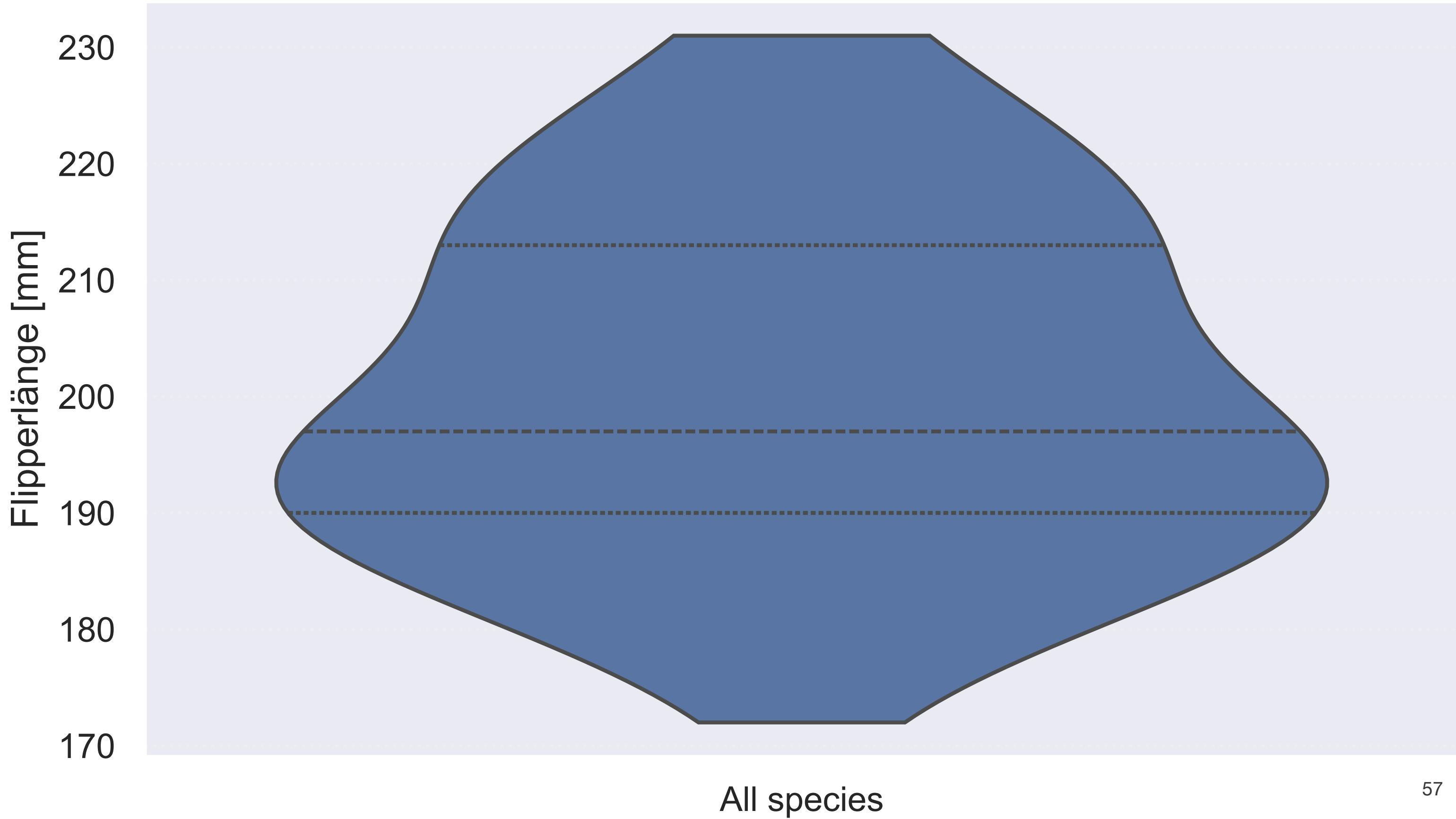
Violinplot – Idee & Lesen

Violin zeigt Dichteform pro Gruppe.

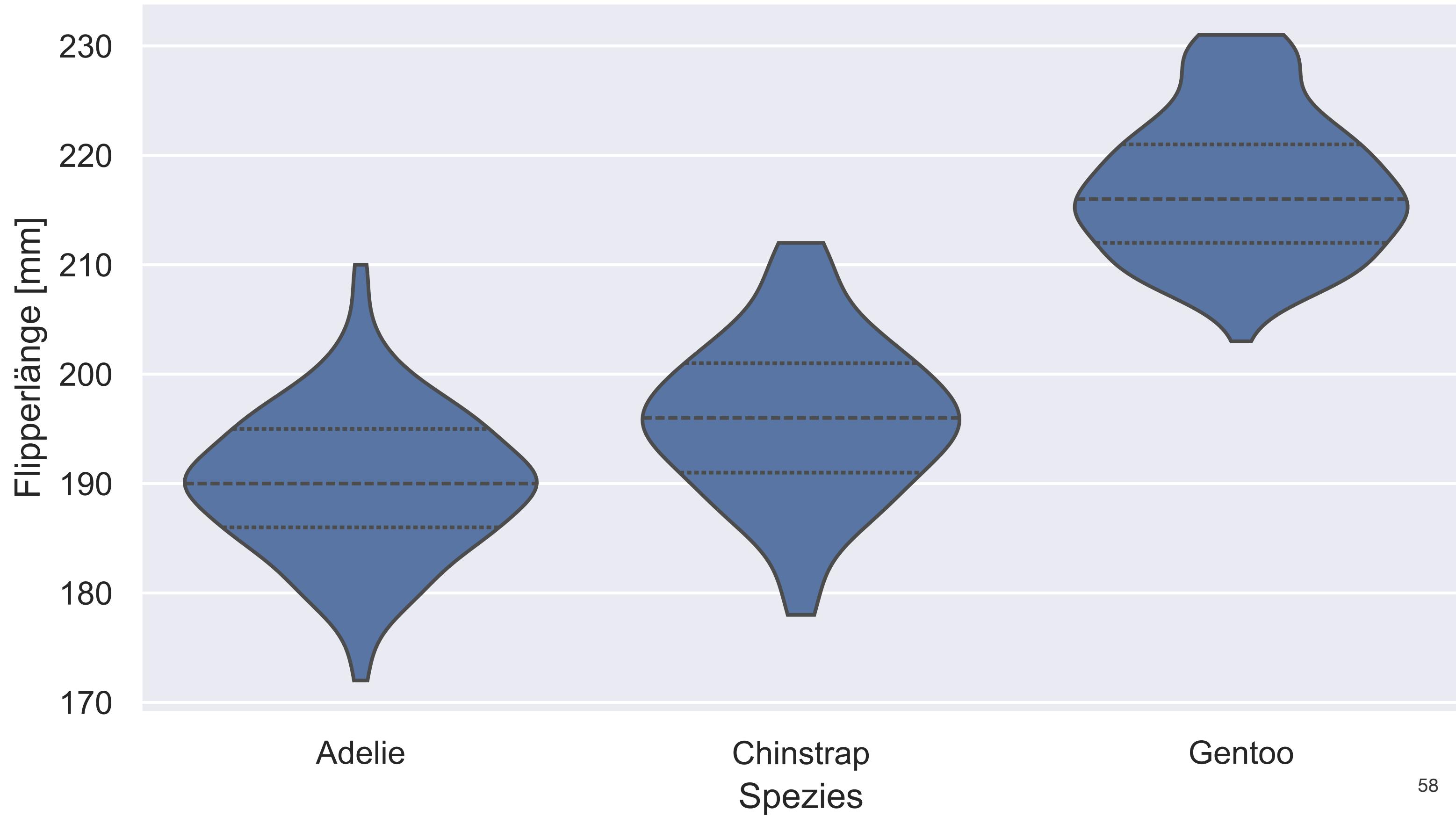
- Basiert auf KDE, Breite \propto Dichte
- Quartile innen darstellbar
- Zeigt Multimodalität besser als Boxplot
- Sensitiv auf Bandbreitenwahl (h)

Mini-Check: Wann ist Violin überlegen?

Penguins – Flipperlänge (alle Spezies) • Violin mit Quartilen



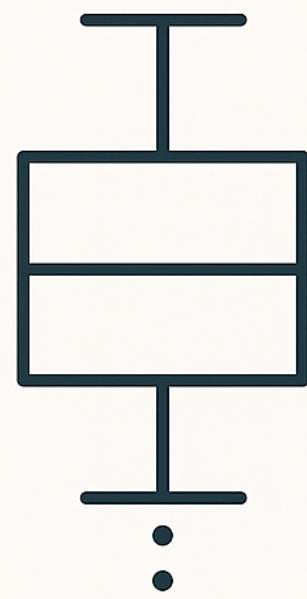
Penguins – Violin mit Quartilen (cut=0)



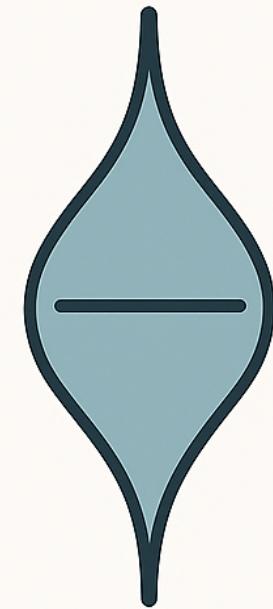
Box vs. Violin

Plotwahl folgt Frage.

- **Boxplot:** robust & kompakt (Median + IQR)
- **Violinplot:** formreich, zeigt Moden & Tails
- Empfehlung:
 - Violin mit Quartilen
 - oder Boxplot + Punkte (Strip/Swarm)
- Einheitliche Skalen für Vergleiche



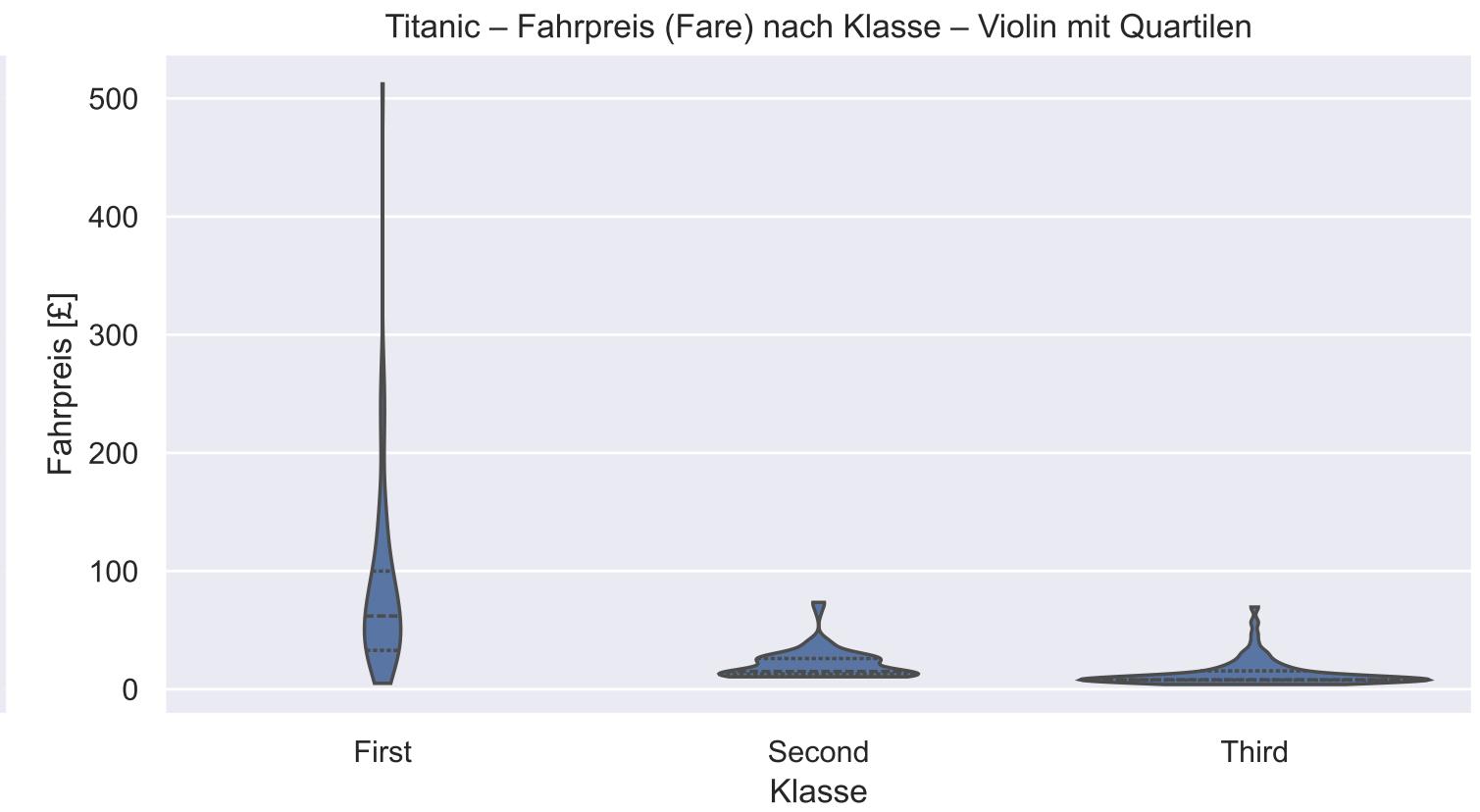
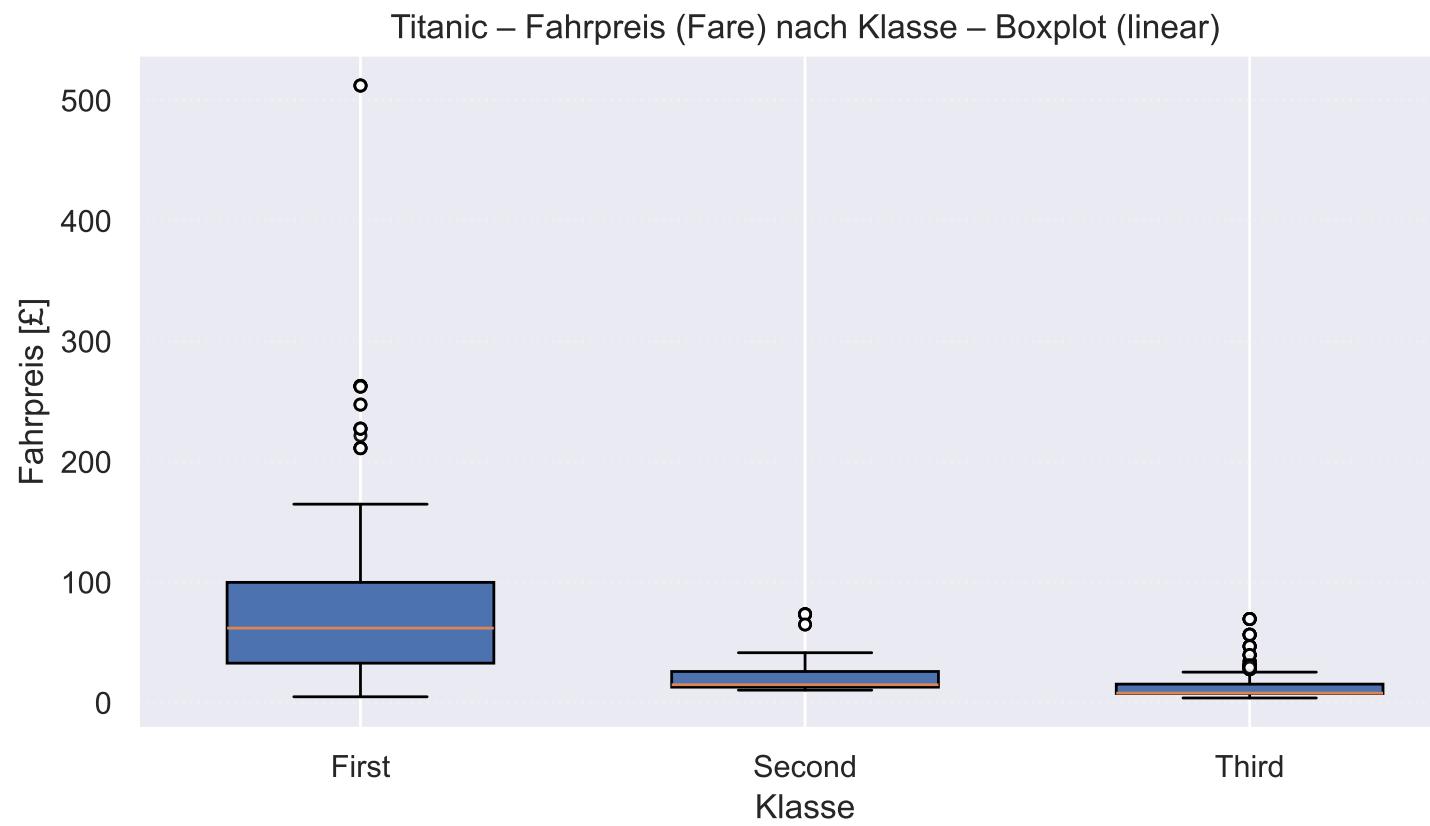
Boxplot



Violinplot

Mini-Check: Welche Kombi bei kleinem n ?

Box vs. Violin



Wahl hängt von Analyseziel ab, Kombination oft ideal

Python Box und Violin

Wenige Parameter genügen.

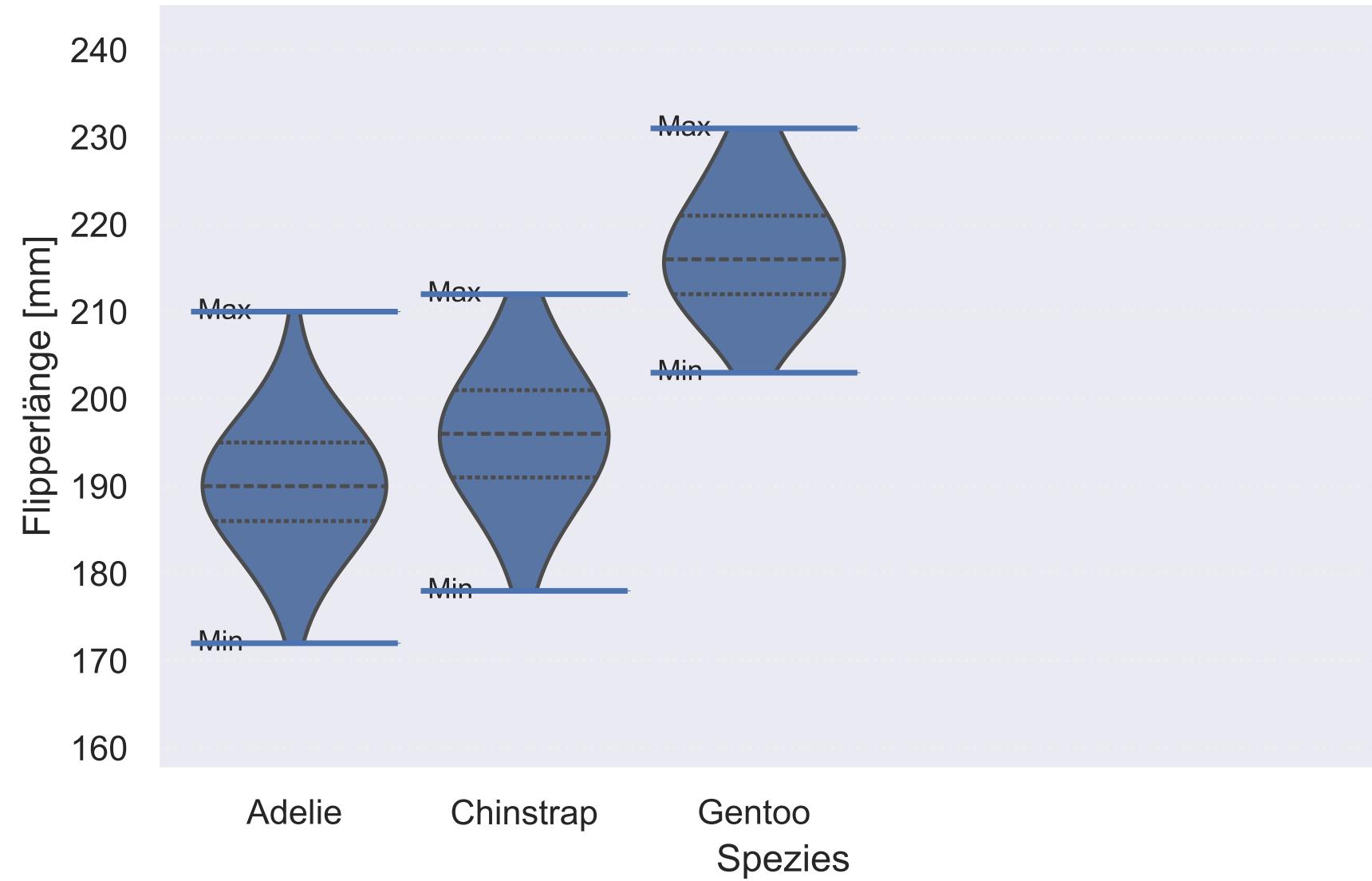
```
sns.boxplot(data=df, x="species", y="flipper_length_mm",
             whis=1.5, showfliers=True)
sns.violinplot(data=df, x="species", y="flipper_length_mm",
                inner="quartile", cut=0, bw_adjust=1.0)
```

- Log-Skala optional: `ax.set_yscale("log")`
- Parameter steuern Aussage (`whis`, `inner`, `cut`, `bw_adjust`)

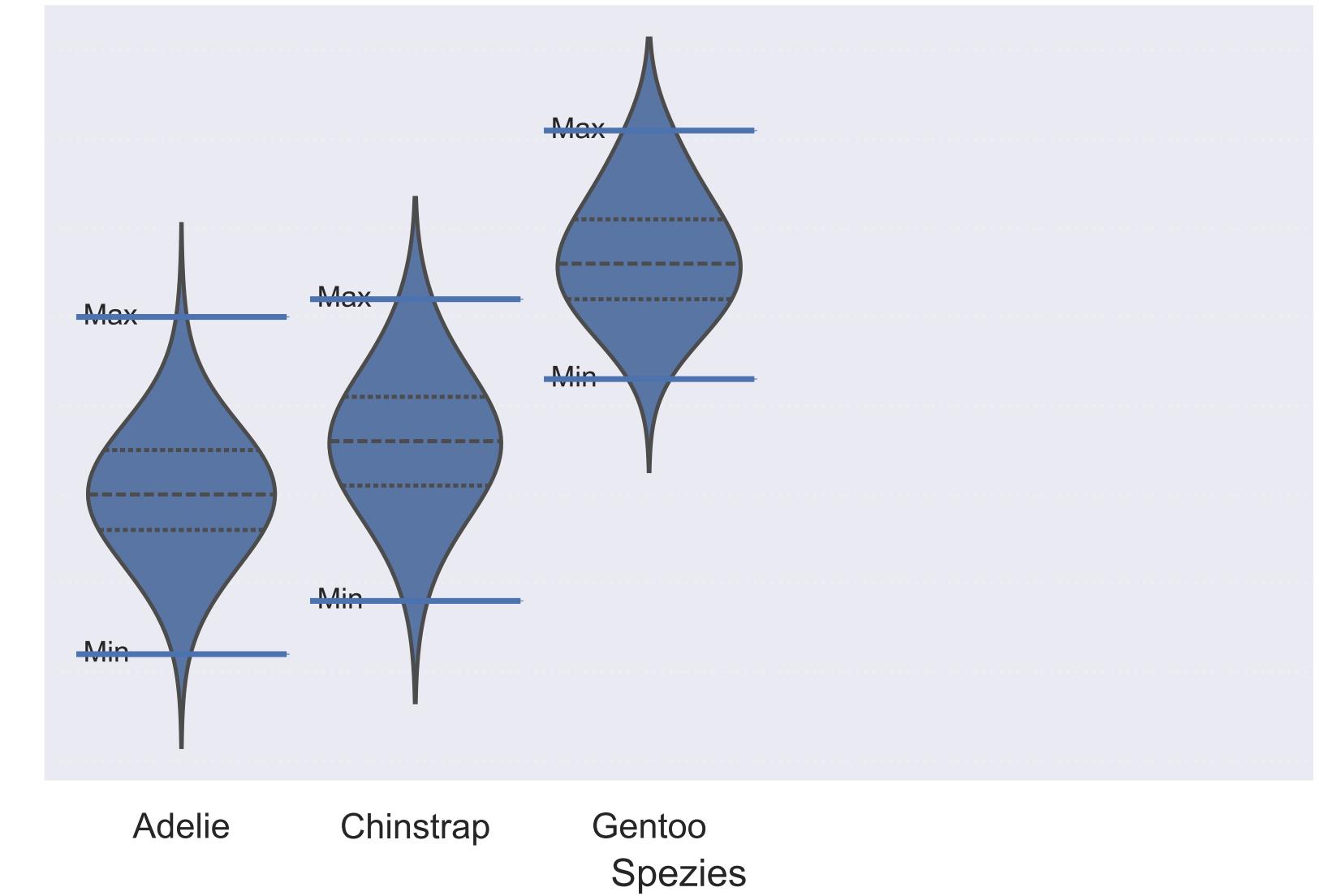
Mini-Check: Warum `cut=0` im Violinplot?

Penguins – Violin: cut=0 vs. Standard (mit Min/Max-Brackets rechts)

Violin (cut=0) – endet am Datenbereich



Violin (cut=2, Standard) – KDE über Min/Max hinaus



Violinplot und die Zipfel

- Violinplots zeigen die **Form der Verteilung** (wie viele Werte wo liegen).
 - Dafür wird eine **glatte Kurve** (KDE) über die Daten gelegt.
 - Diese Kurve rutscht manchmal **über die echten Daten hinaus** → es entstehen **Zipfel**, sog. Rand-Effekte (boundary effects)
 - Beispiel: kleinste Flipperlänge = 170 mm → Kurve geht trotzdem bis 150 mm.
 - Grund: die Glocke hört nie ganz auf (wie beim Normalverteilten).
- 👉 Für **Analysen** ist das ok – man sieht die Kurvenform schön.
- 👉 Für **Präsentationen** oft besser: `cut=0` → Kurve stoppt bei den echten Daten.

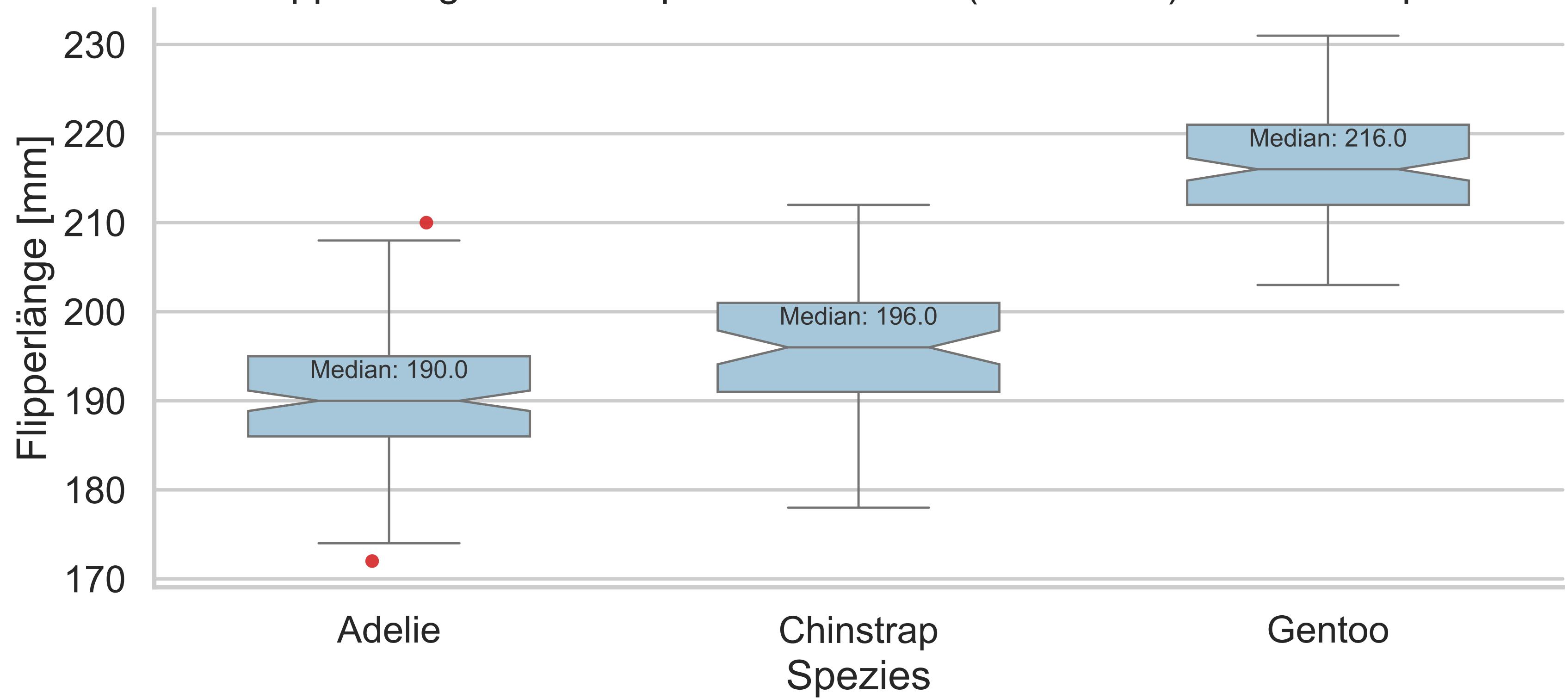
Gruppenvergleich – Boxplot

Robuste Gruppenvergleiche zuerst ansehen.

- Mediane zwischen Gruppen vergleichen
- IQR-Überlappung prüfen
- Ausreisser getrennt betrachten
- **Textbaustein:** Median A > Median B

Mini-Check: Woran erkennst du echten Medianunterschied?

Gruppenvergleich – Boxplot mit Notches (Median-CI) + Outlier separat



Gruppenvergleich – Violinplot

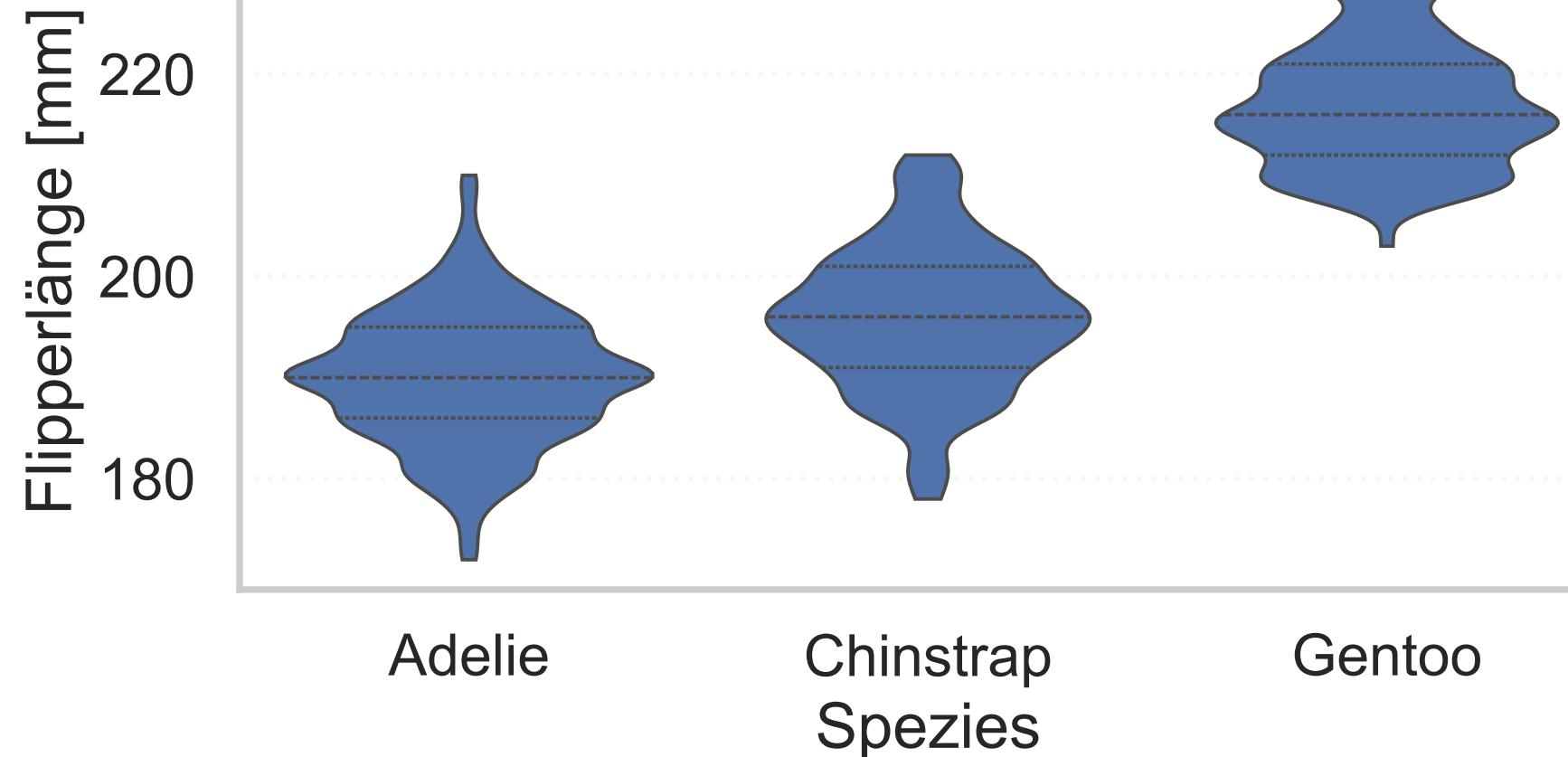
Formunterschiede sichtbar machen.

- Modenverschiebung und Tail-Unterschiede erkennbar
- Quartile innen als Orientierung
- Bandbreite (bw_adjust) variieren

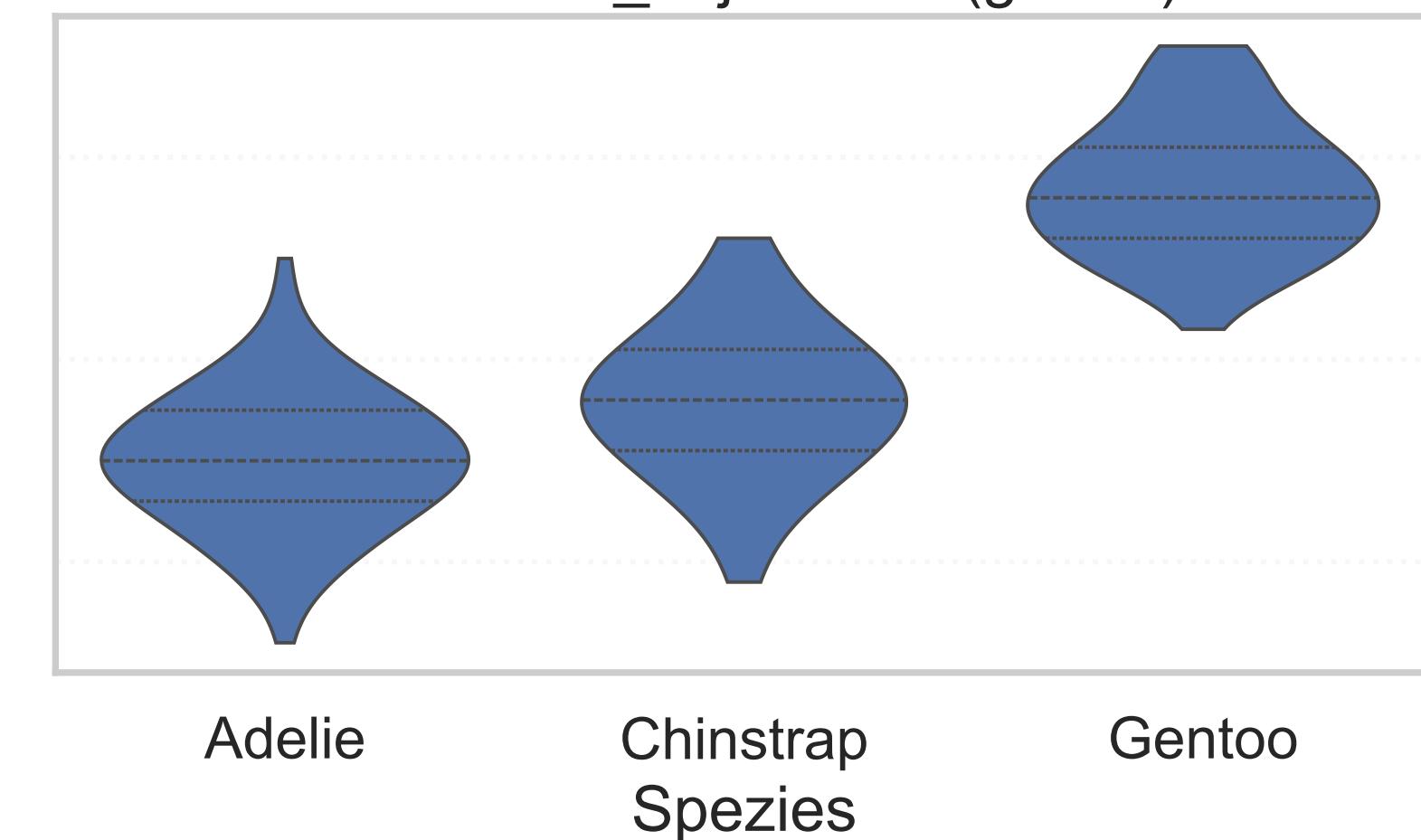
Mini-Check: Welche Form spricht für Mischungen in den Daten?

Gruppenvergleich – Violinplot (Quartile innen, cut=0, Bandbreite variieren)

Violin – bw_adjust=0.6 (mehr Details)



Violin – bw_adjust=1.4 (glatter)



Punkte ergänzen Strip oder Swarm

Rohdaten vermeiden Fehlinterpretationen.

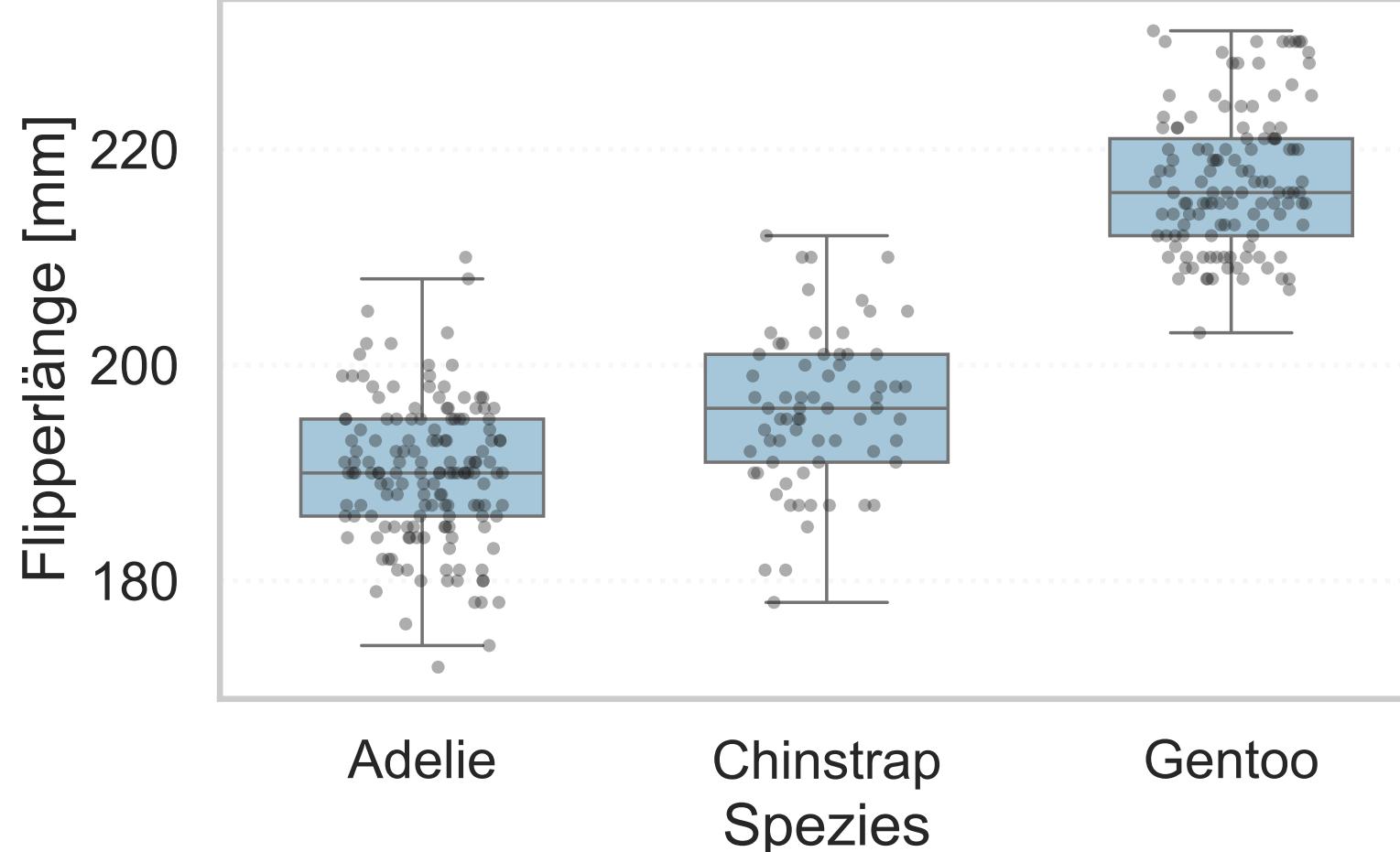
```
sns.boxplot(..., showfliers=False)
sns.stripplot(..., color="k", alpha=0.35, jitter=0.2)
# oder:
sns.swarmplot(..., color="k", alpha=0.35)
```

- Rohdatenpunkte machen Verteilung sichtbar
- Überdeckung per Transparenz mindern
- Kombination mit Boxplot = robuste Kennzahlen + echte Daten

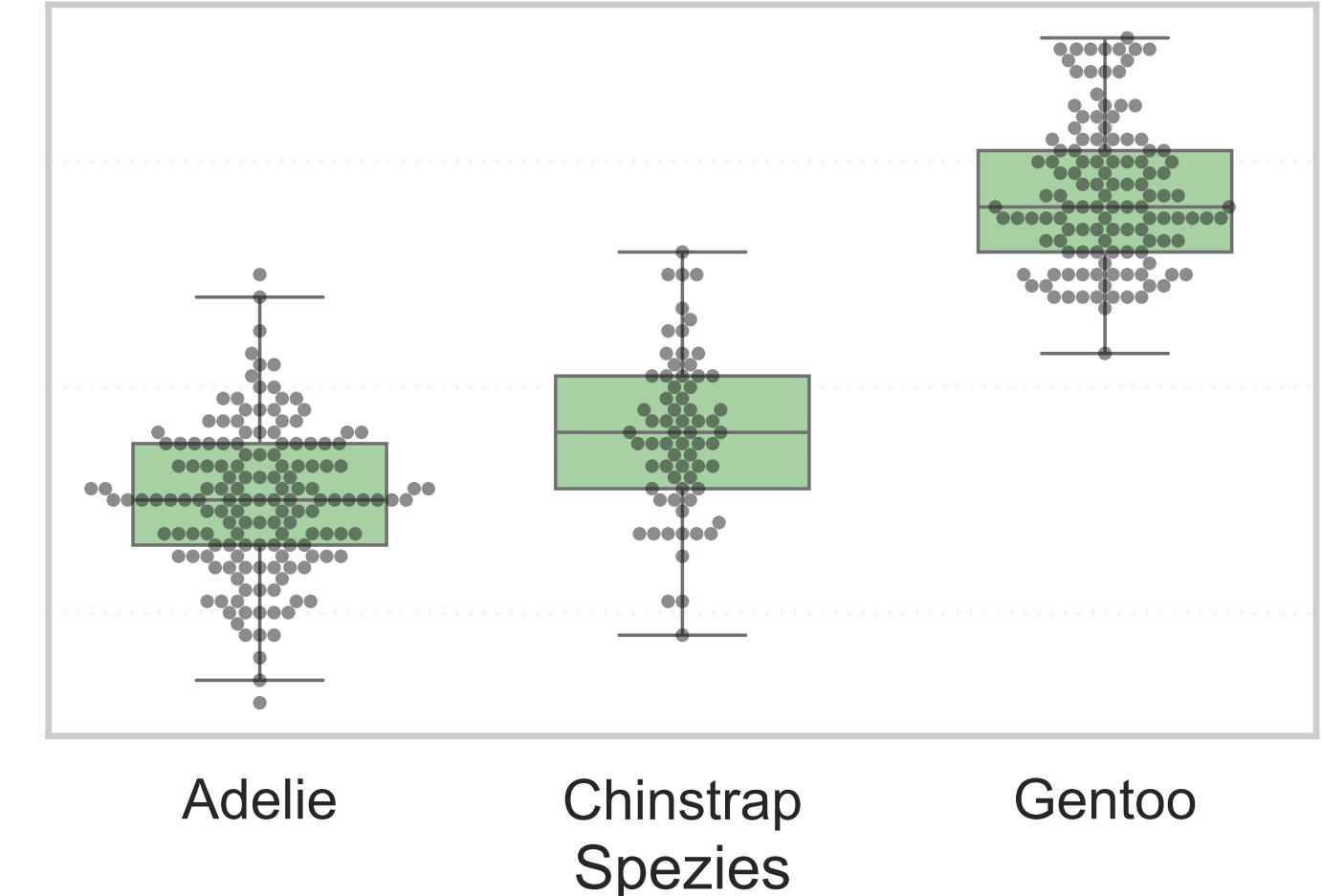
Mini-Check: Warum showfliers=False beim Overlay?

Punkte ergänzen Strip oder Swarm – Rohdaten sichtbar machen

Boxplot + Stripplot (Rohdaten + Kennzahlen)

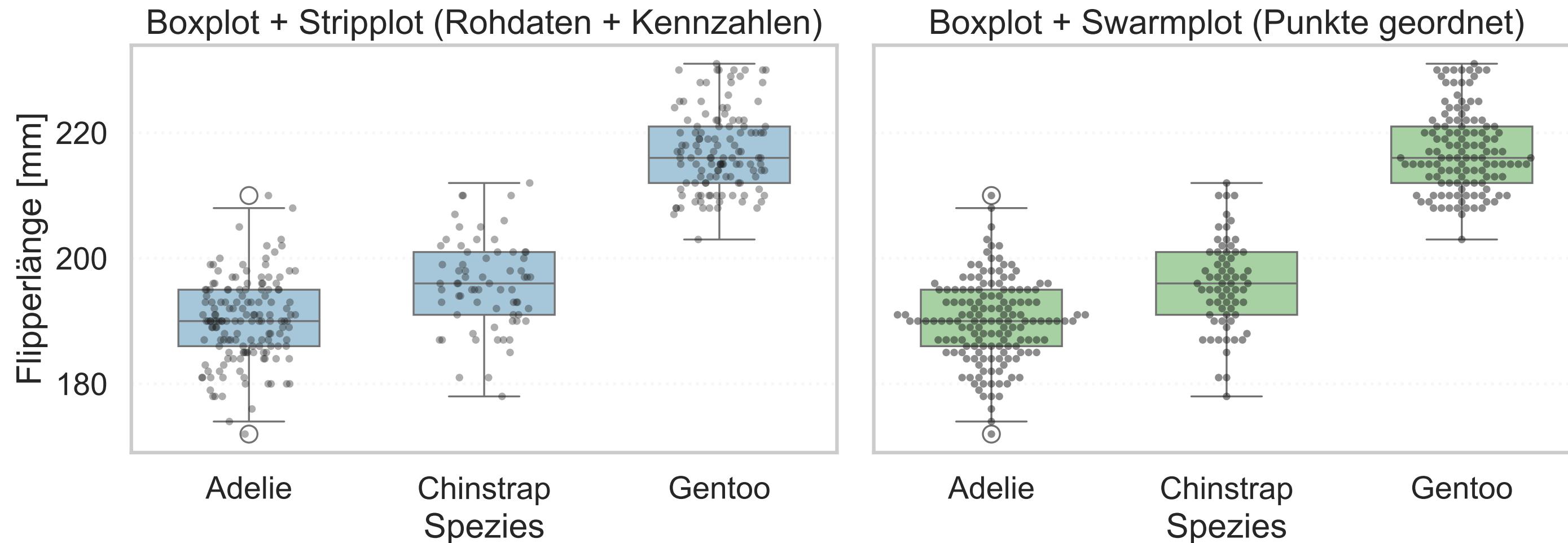


Boxplot + Swarmplot (Punkte geordnet)



Mit `showfliers=true`

Punkte ergänzen Strip oder Swarm – Rohdaten sichtbar machen



Strip vs. Swarm

Strip (sns.stripplot)

- Punkte mit optionalem Jitter
- Überlappungen möglich
- Sehr schnell, robust bei grossem **n**
- Gut als Overlay zu Boxplot
- Tipps: `alpha=0.35,`
`jitter=0.2, dodge=True` bei Hue

Wann? Viele Punkte, schnelle Vorschau, Fokus auf Rohdaten

Swarm (sns.swarmplot)

- Bienenwaben: kollisionsfrei entlang x
- Lokale Dichte besser sichtbar
- Langsamer, bei sehr grossem **n** überladen
- Gut als Stand-alone oder Overlay

Wann? Kleines bis mittleres **n**, Formdetails wichtig

Häufige Fehlinterpretationen

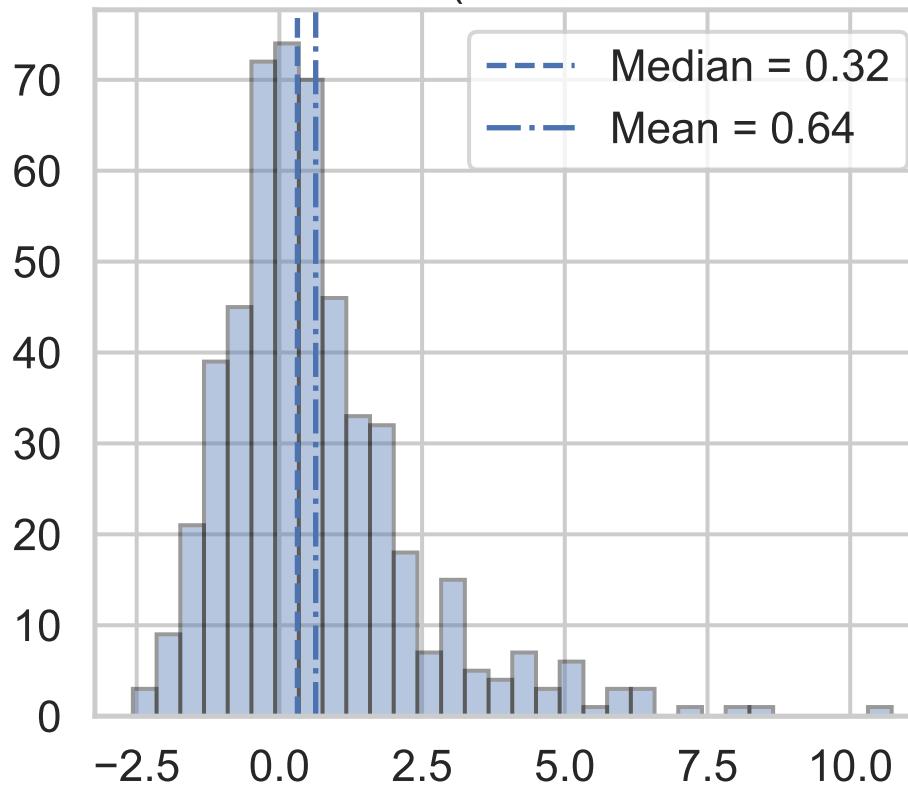
Fallen kennen und vermeiden.

- Median \neq Mittelwert
- Boxhöhe \neq Standardabweichung
- Whisker \neq Min/Max (bei `whis=1.5`)
- Log-Skala muss klar gekennzeichnet sein

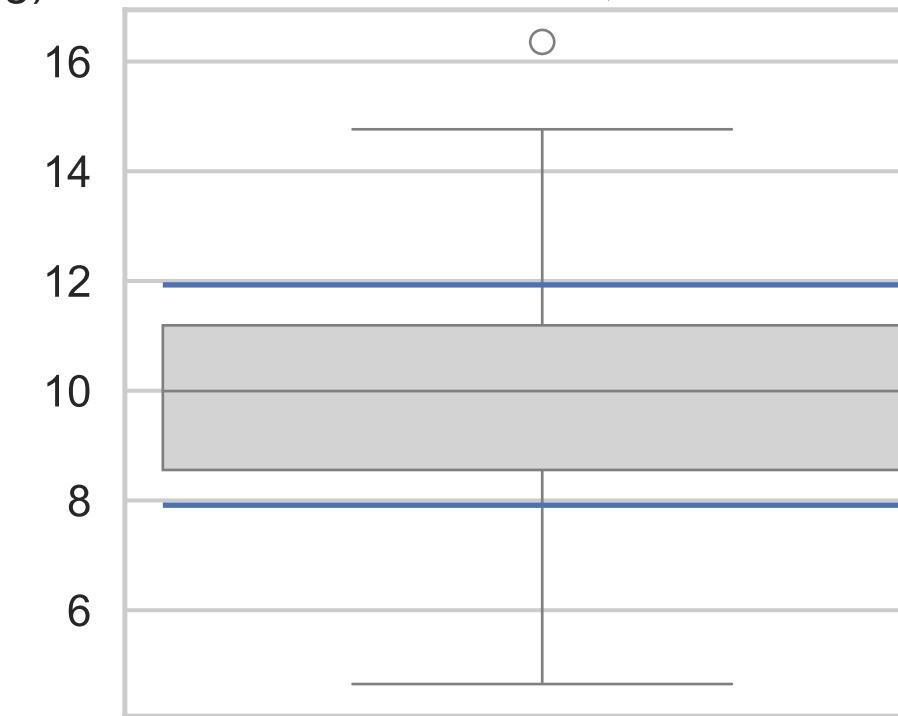
Mini-Check: Wie markierst du n je Gruppe?

Häufige Fehlinterpretationen – Mini-Plots

Median \neq Mittelwert (rechts-schiefe Verteilung)

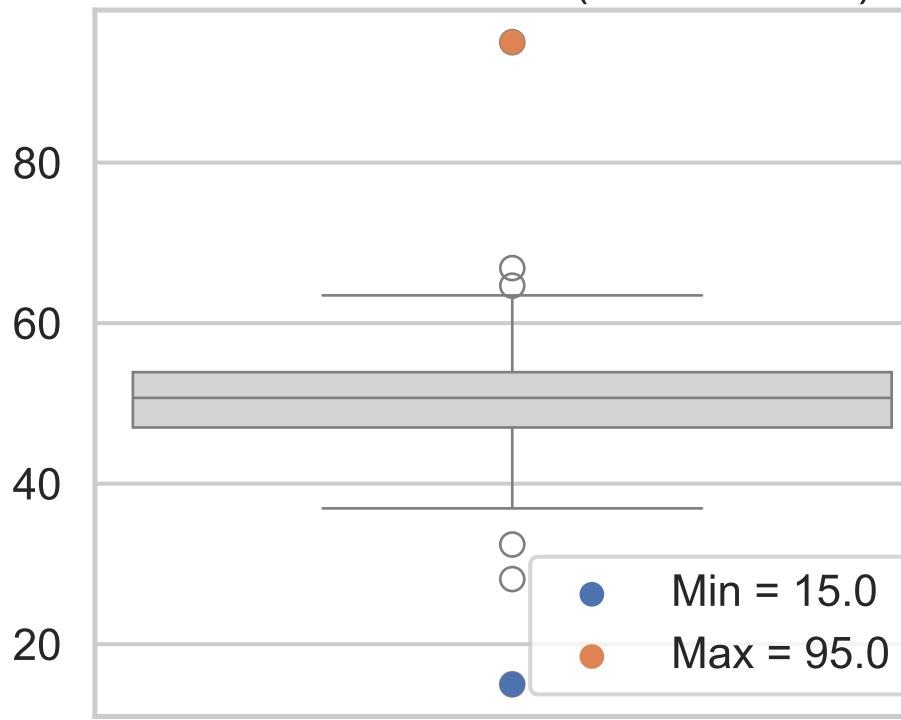


Boxhöhe = IQR, nicht SD

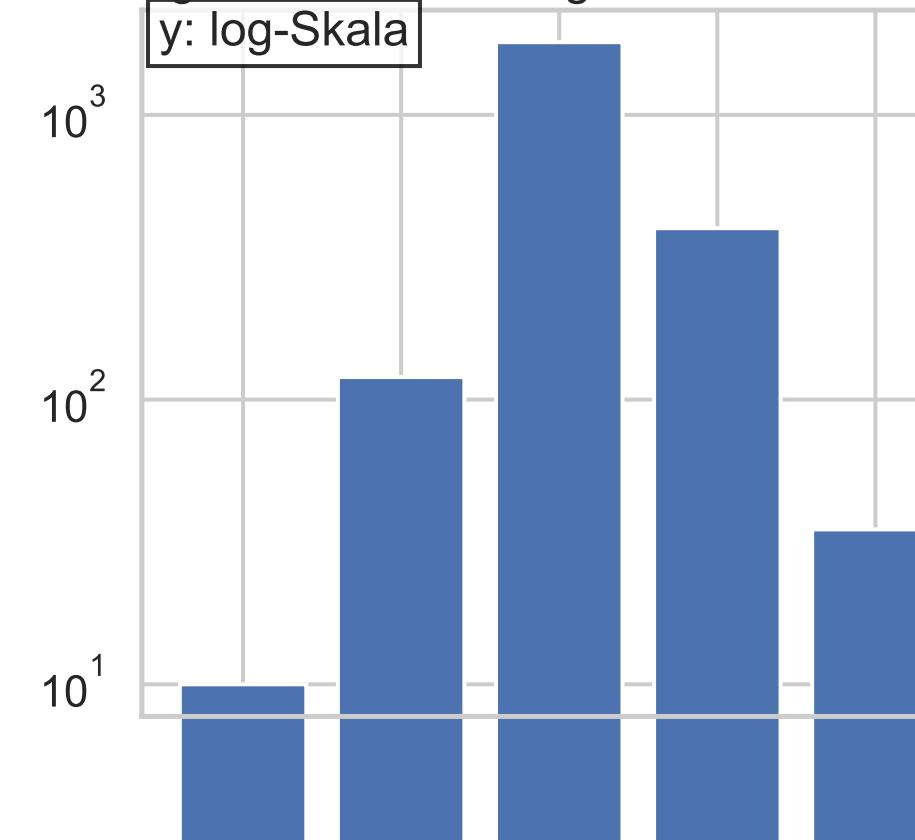


±1 SD IQR

Whisker \neq Min/Max (bei whis=1.5)



Log-Skala muss klar gekennzeichnet sein

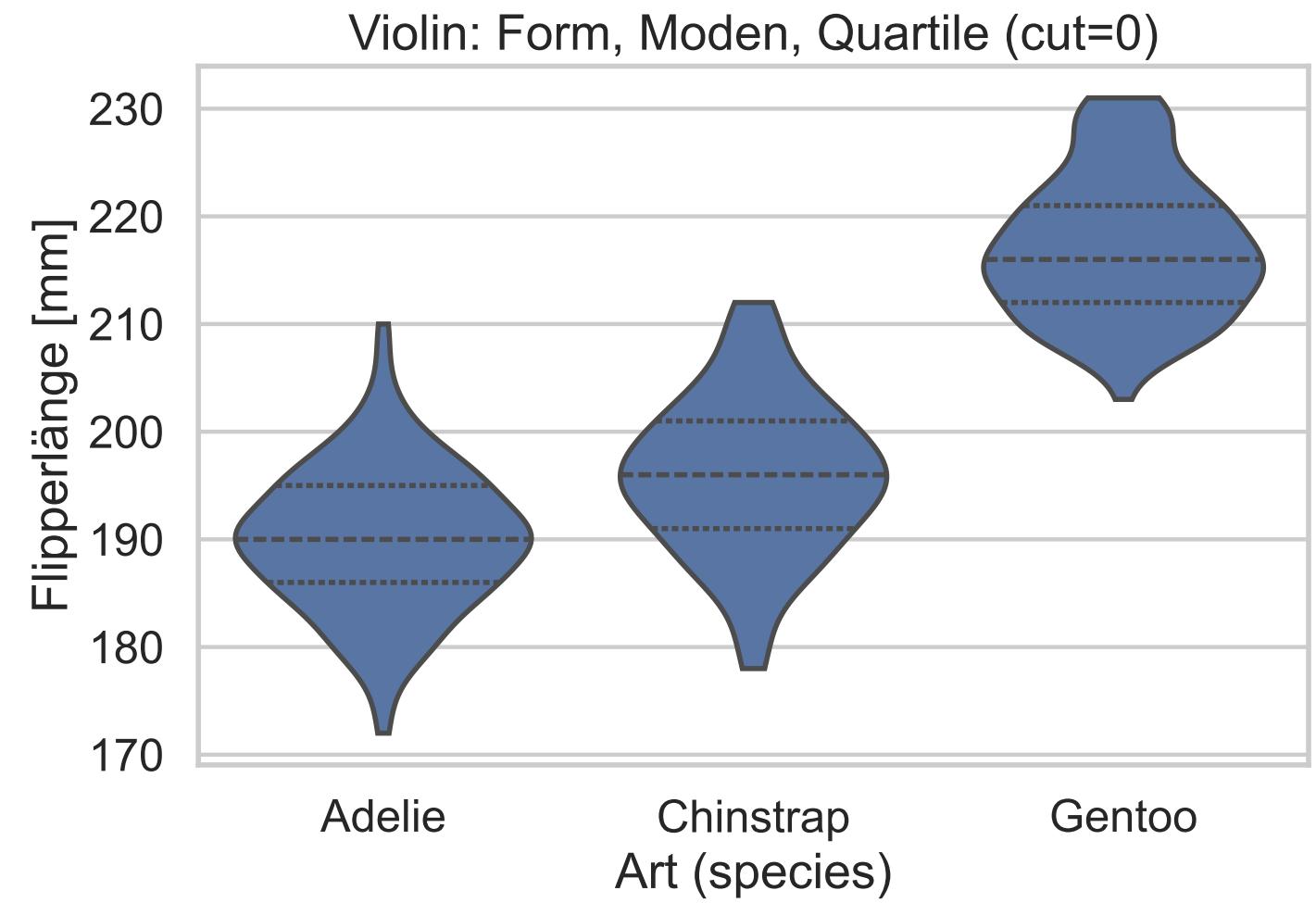
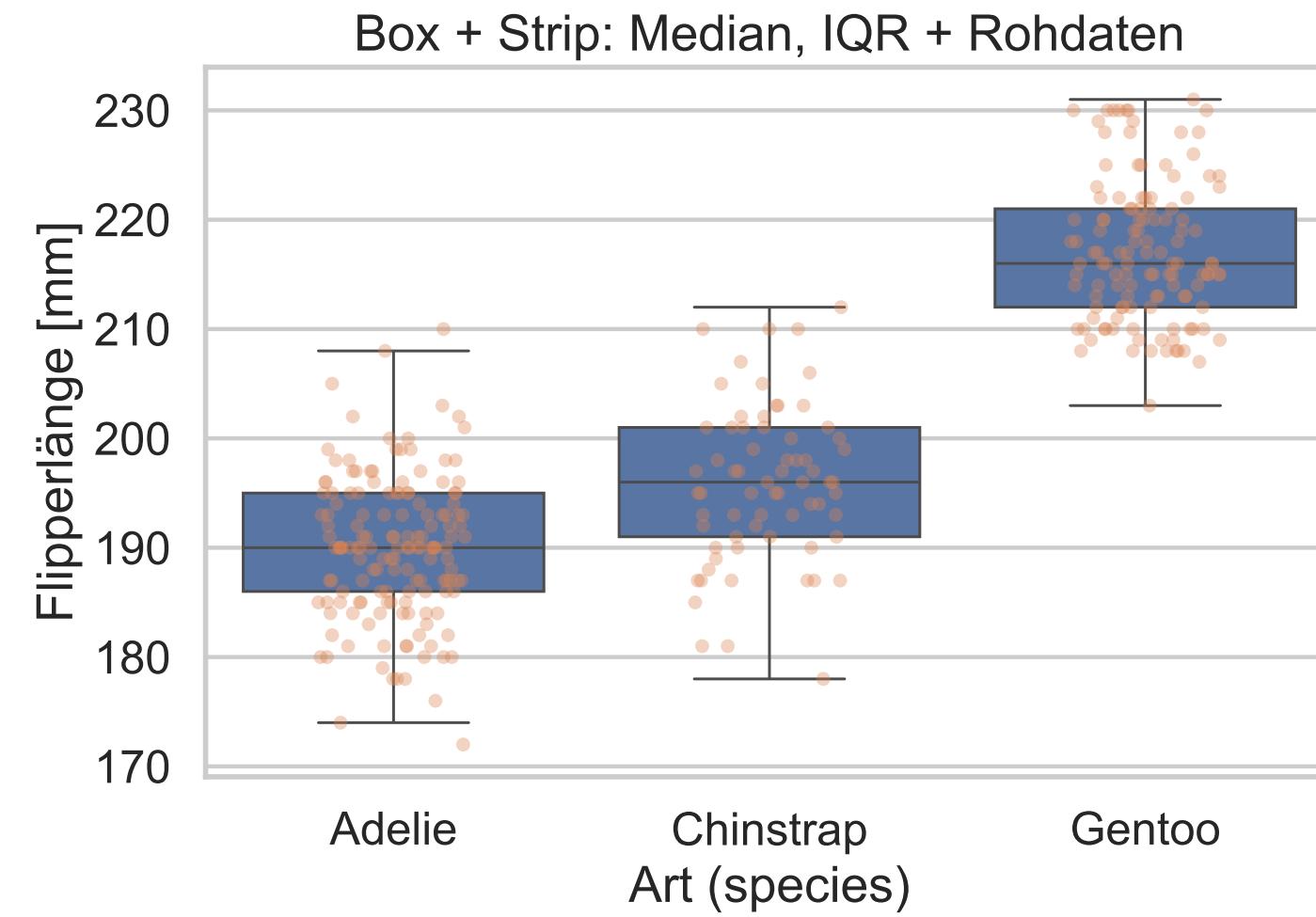


Mini-Case: Unterschied oder Ausreißer?

Entscheidung mit Plotkombi und Kennzahlen.

- Box + Strip prüfen → zeigen Median, IQR und Rohdaten
- Violin für Form und Moden
- Kennzahlen vergleichen: Mediane, IQR, ggf. MAD
- Kurze Dokumentation der Entscheidung

Mini-Check: 2-Satz-Entscheidung formulieren



Kennzahlen je Gruppe (Median, IQR, MAD, n, Outlier-Counts)

	n	Median	Q1	Q3	IQR	MAD	Tukey_outlier	modZ_outliers
Adelie	151	190.0	186.0	195.0	9.0	5.0	2	0
Chinstrap	68	196.0	191.0	201.0	10.0	5.0	0	0
Gentoo	123	216.0	212.0	221.0	9.0	4.0	0	0

Key Takeaways – Box & Violin

- **Boxplot:** kompakt, robust → Median, IQR, Ausreisser
 - **Varianten:** whis, Notches, Log-Skala, Gruppen nebeneinander
 - **Violinplot:** zeigt Verteilungsform & Multimodalität, sensibel für Bandbreite
 - **Kombination:** Box + Punkte oder Violin + Quartile → mehr Aussagekraft
 - **Praxis:** Einheitliche Skalen & Gruppengrößen (n) immer angeben
- 👉 Robustheitsmasse + Formdarstellung ergeben den vollen Überblick.

ECDF und QQ

ECDF (Empirical Cumulative Distribution Function - Empirische Verteilungsfunktion)

QQ (Quantile-Quantile - Quantil-Quantil-Diagramm)

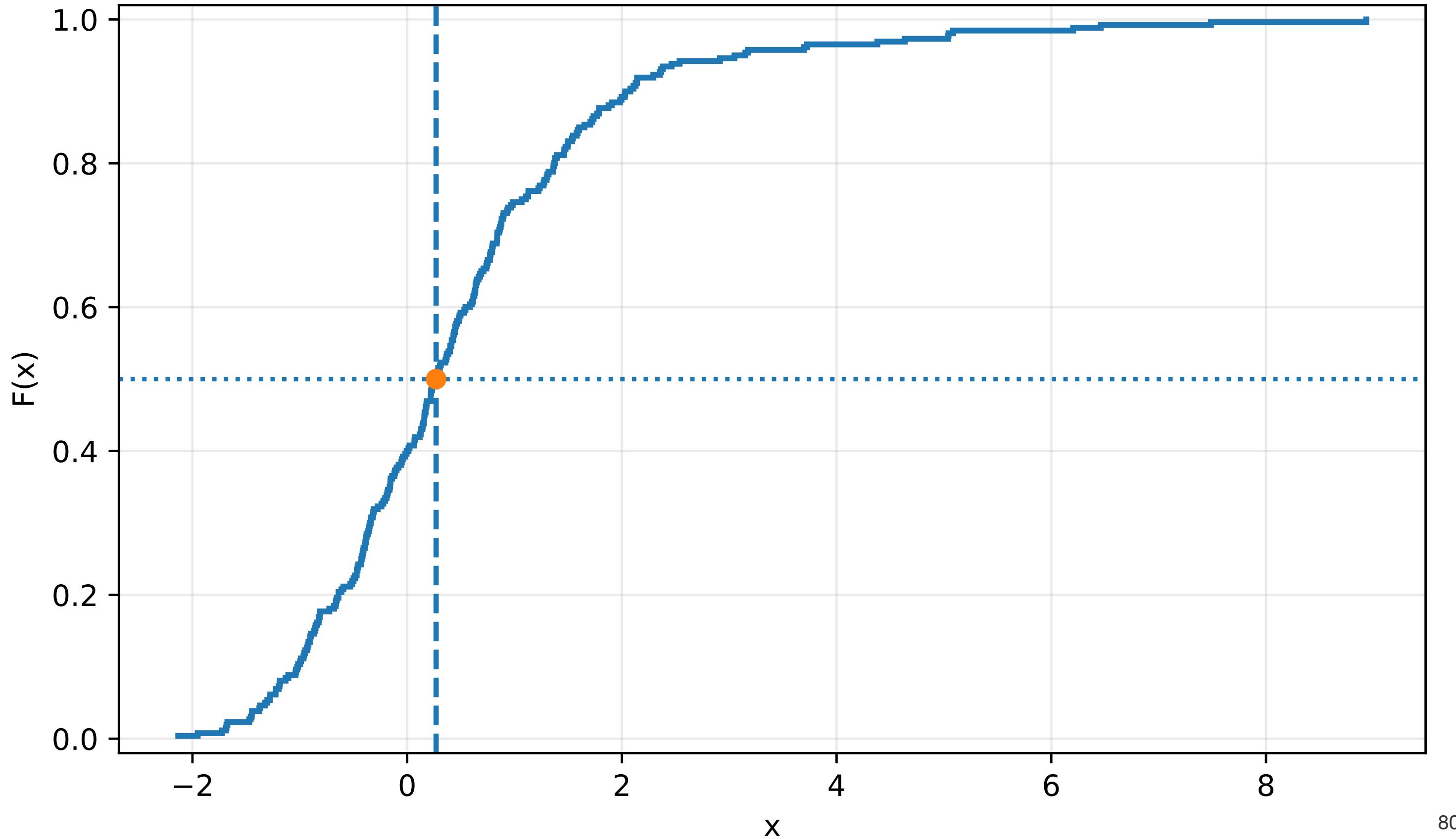
ECDF – Grundidee

ECDF ist binfrei und vollständig.

- Empirical Cumulative Distribution Function (ECDF)
- $\hat{F}(x) = \frac{1}{n} \sum \mathbf{1}\{x_i \leq x\}$
- Monoton steigend von 0 bis 1
- Quantile direkt ablesbar
- Besonders nützlich bei kleinem n und Schiefe

Mini-Check: Warum oft informativer als grobes Histogramm?

ECDF (binfrei) mit markiertem Median



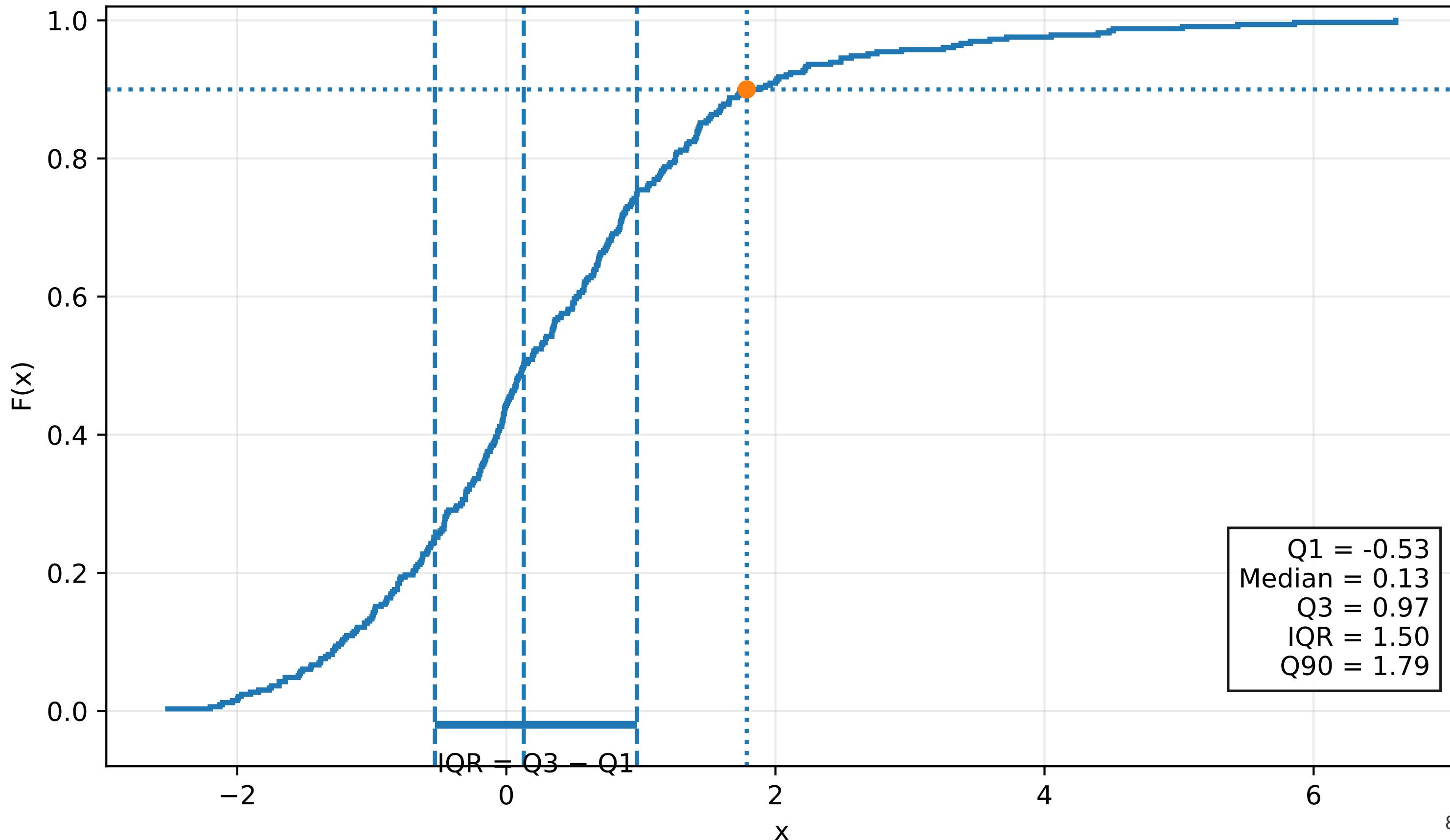
ECDF – Lesen & Quantile

Median und Quartile direkt lesen

- Median = Punkt, wo $F(x) = 0.5$
- Q1 bei $F(x) = 0.25$, Q3 bei $F(x) = 0.75$
- $IQR = Q3 - Q1$ (auf x-Achse abtragbar)
- Quantile direkt aus Kurve ablesbar

Mini-Check: Wo liegt das 90-Prozent-Quantil?

ECDF mit Quantilen (Q1, Median, Q3) und 90%-Quantil



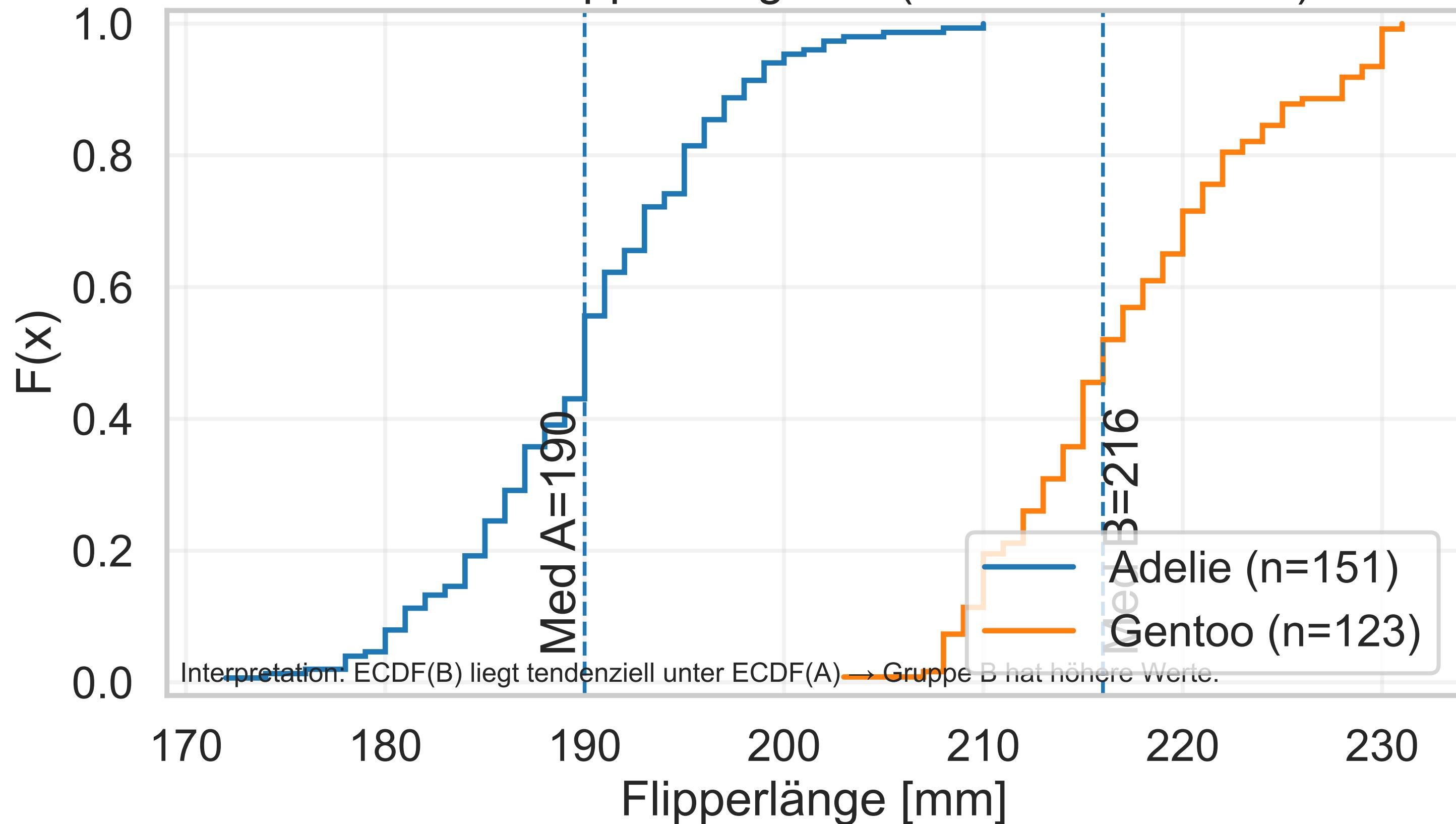
ECDF – Gruppenvergleich

Transparente Vergleiche ohne Binning.

- Zwei oder mehr ECDFs überlagern
- Rechtsverschiebung → höhere Werte in dieser Gruppe
- Kreuzungen → Unterschiede in der Form der Verteilung
- Kennzahlen (Median, IQR) ergänzen

Mini-Check: Was bedeutet ECDF A unter B?

ECDF – Gruppenvergleich (Adelie vs. Gentoo)

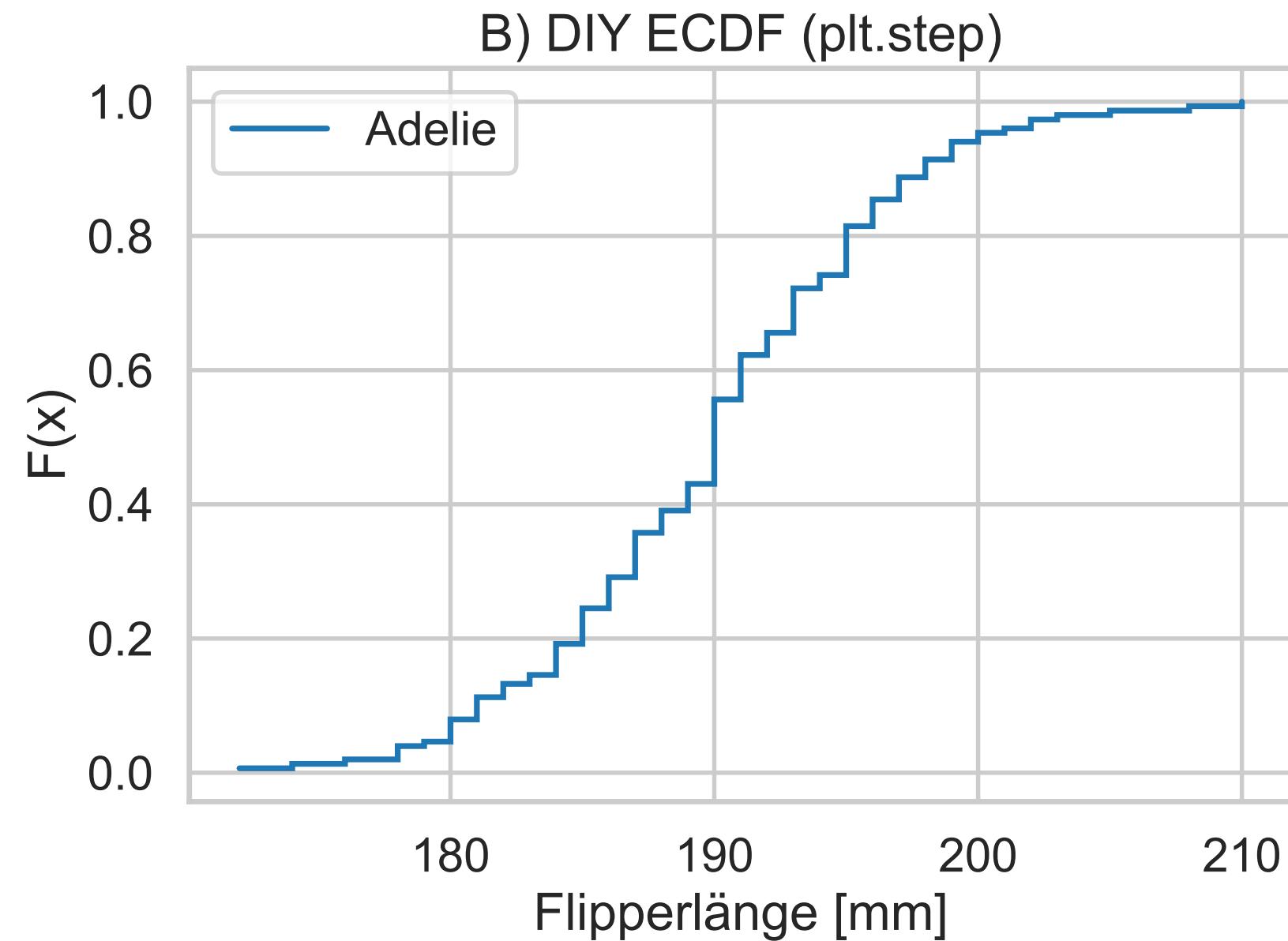
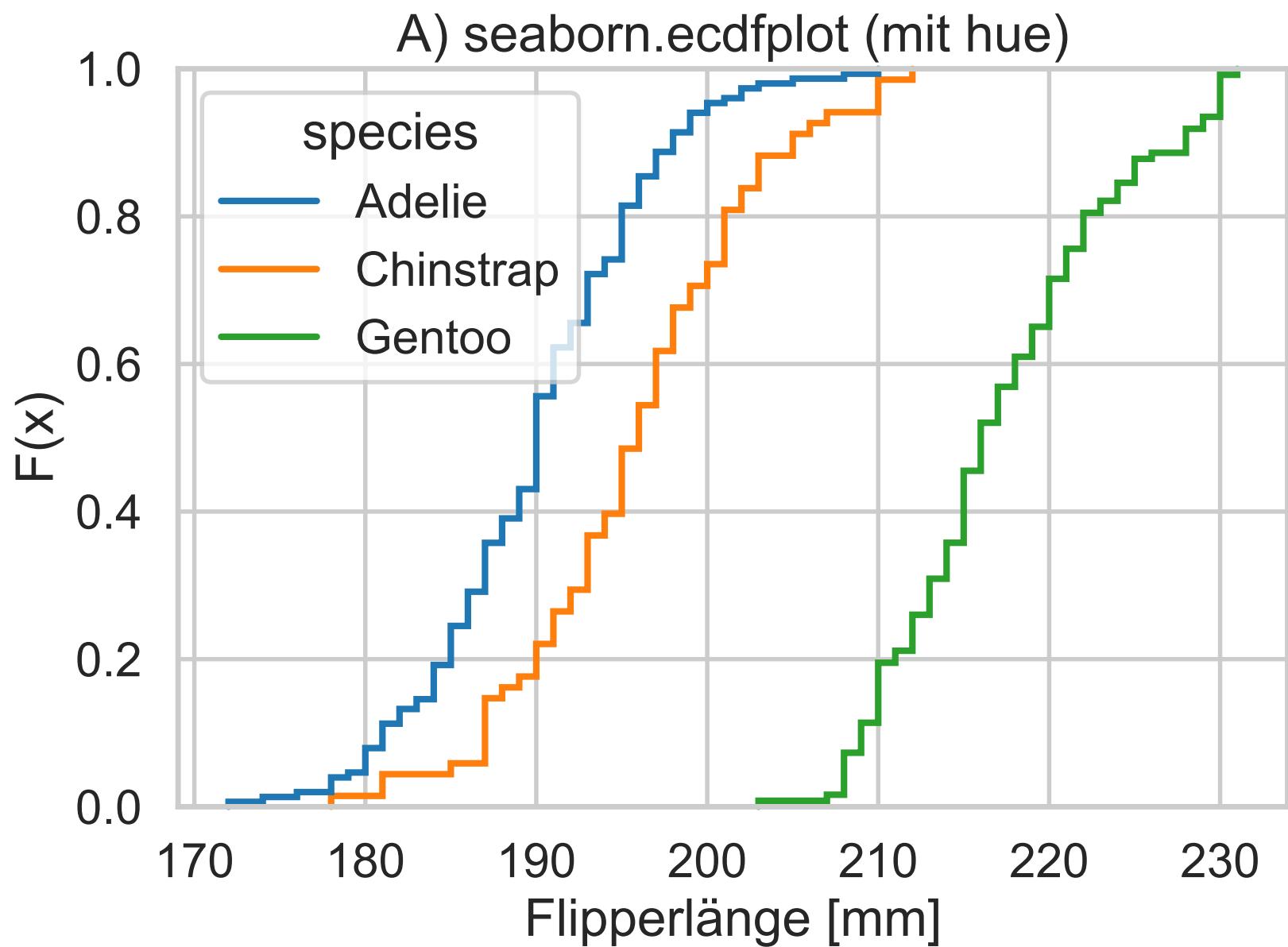


Python ECDF

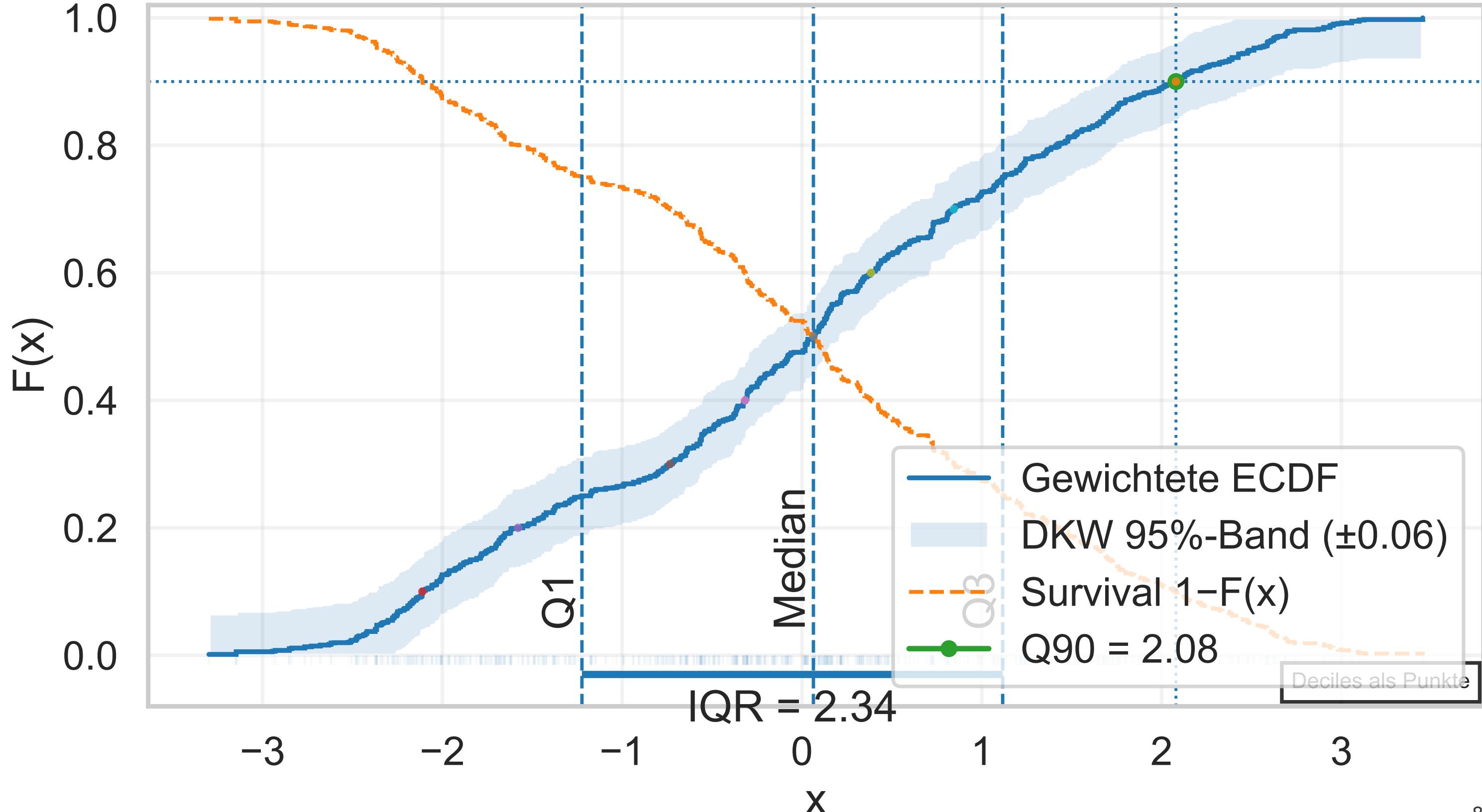
Mit seaborn oder Step-Plot schnell umgesetzt.

```
sns.ecdfplot(data=df, x="flipper_length_mm")
sns.ecdfplot(data=df, x="flipper_length_mm", hue="species")
# DIY:
x = np.sort(s); y = np.arange(1,len(x)+1)/len(x)
plt.step(x, y, where="post")
```

Mini-Check: Wann DIY (Do It Yourself) statt seaborn?



DIY ECDF (gewichtet, post) – DKW-Band, Survival, Quantile



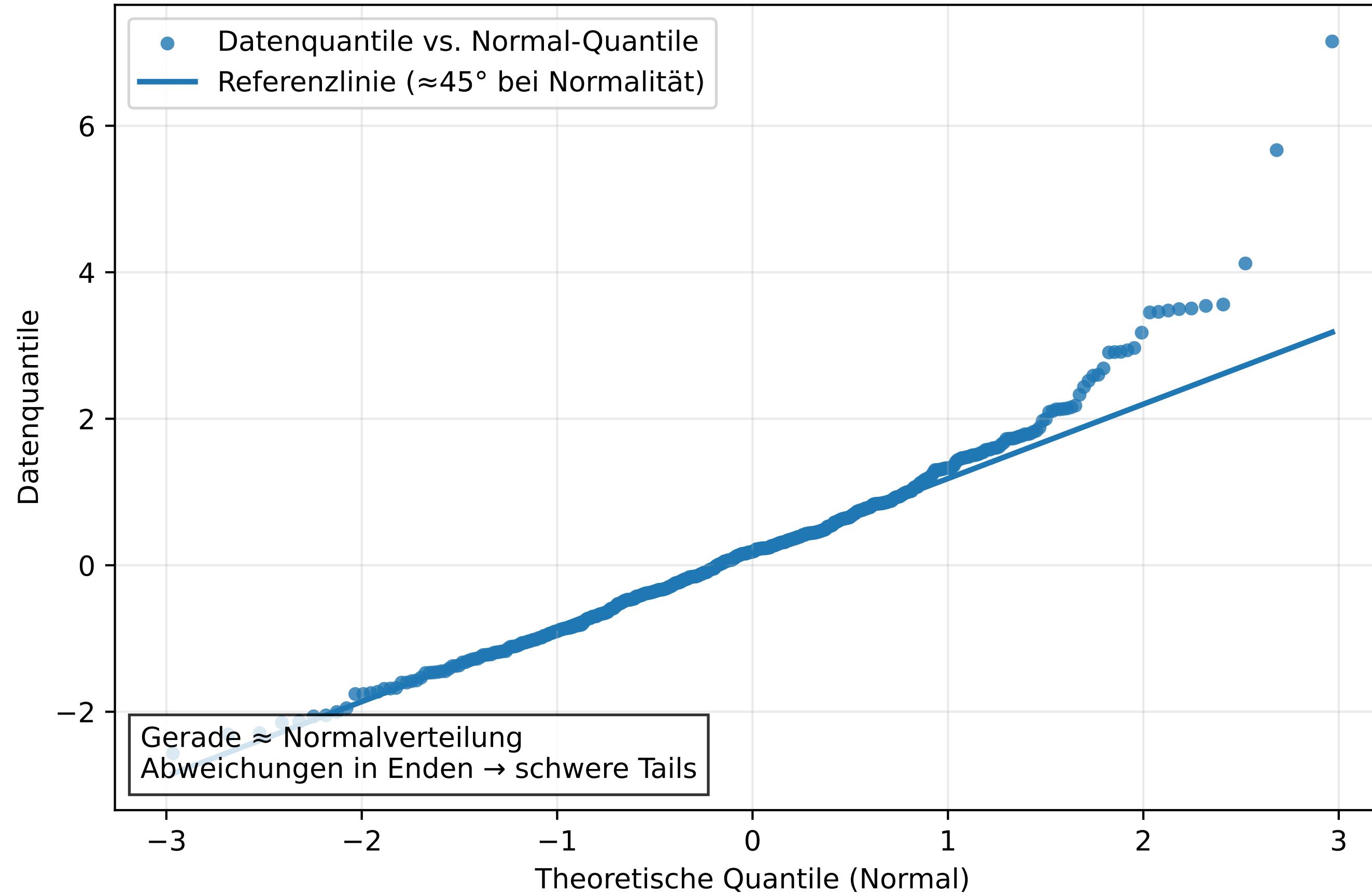
QQ Plot – Grundidee

Datenquantile gegen Referenzquantile.

- Typisch: Referenz = Normalverteilung
- Paare (Q_p^{Daten}, Q_p^{Ref}) → aufgetragen als Punkte
- Gerade Linie = gute Übereinstimmung
- Abweichungen = Hinweise auf Schiefe oder Tails

Mini-Check: Was prüfst du mit Normal-QQ??

QQ-Plot (Normal-Referenz)



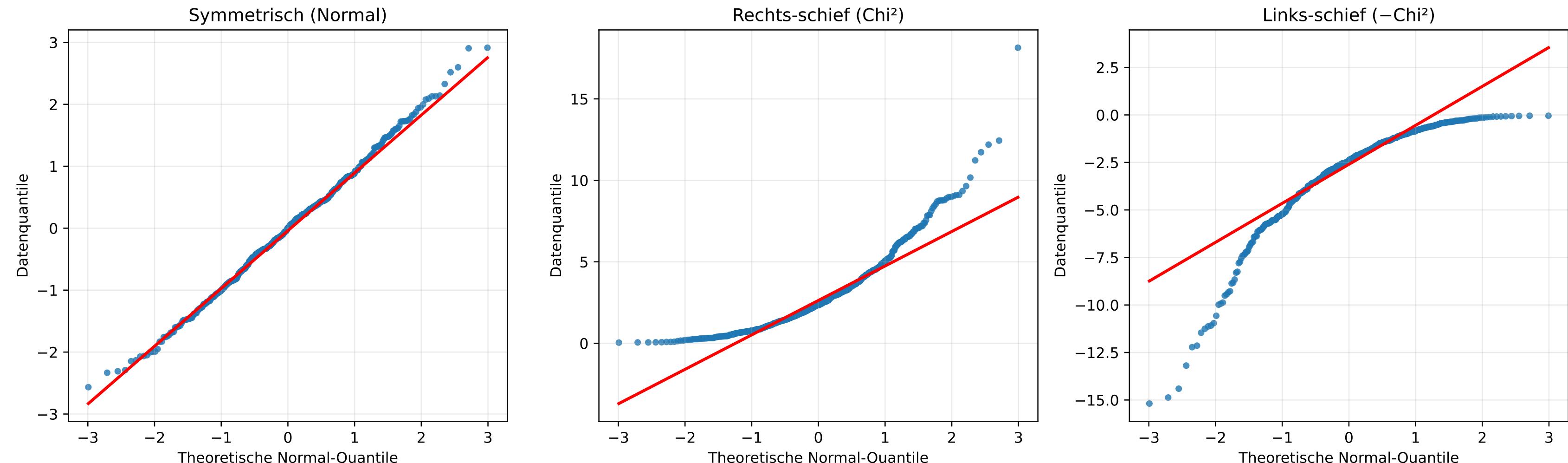
Normal-QQ – Lesen

Abweichungen verraten Schiefe und Tails.

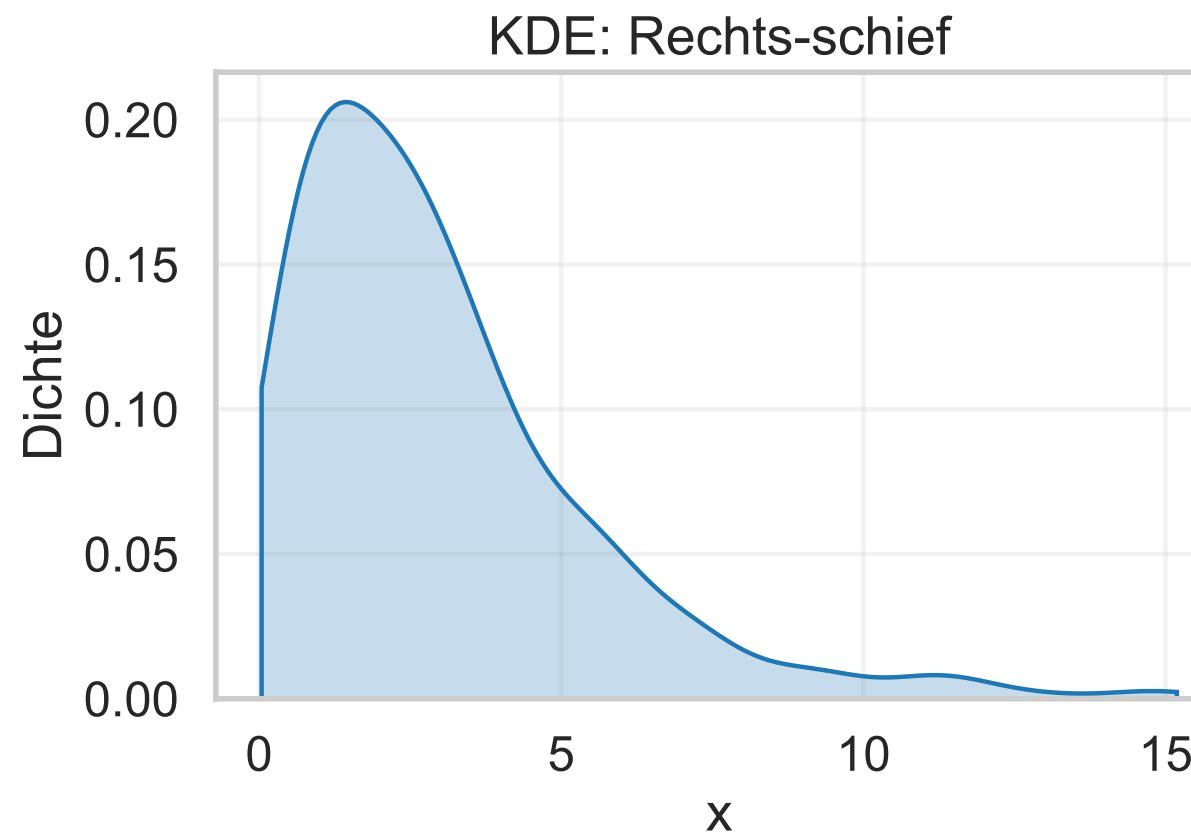
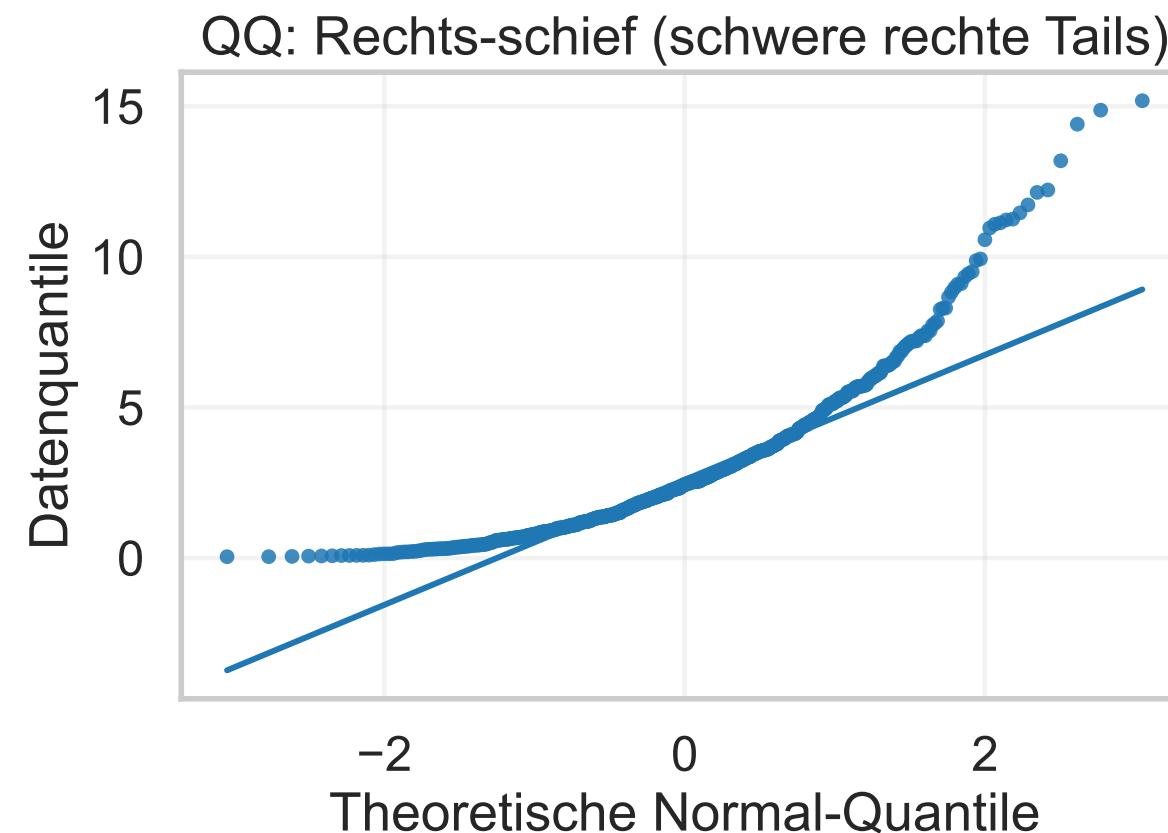
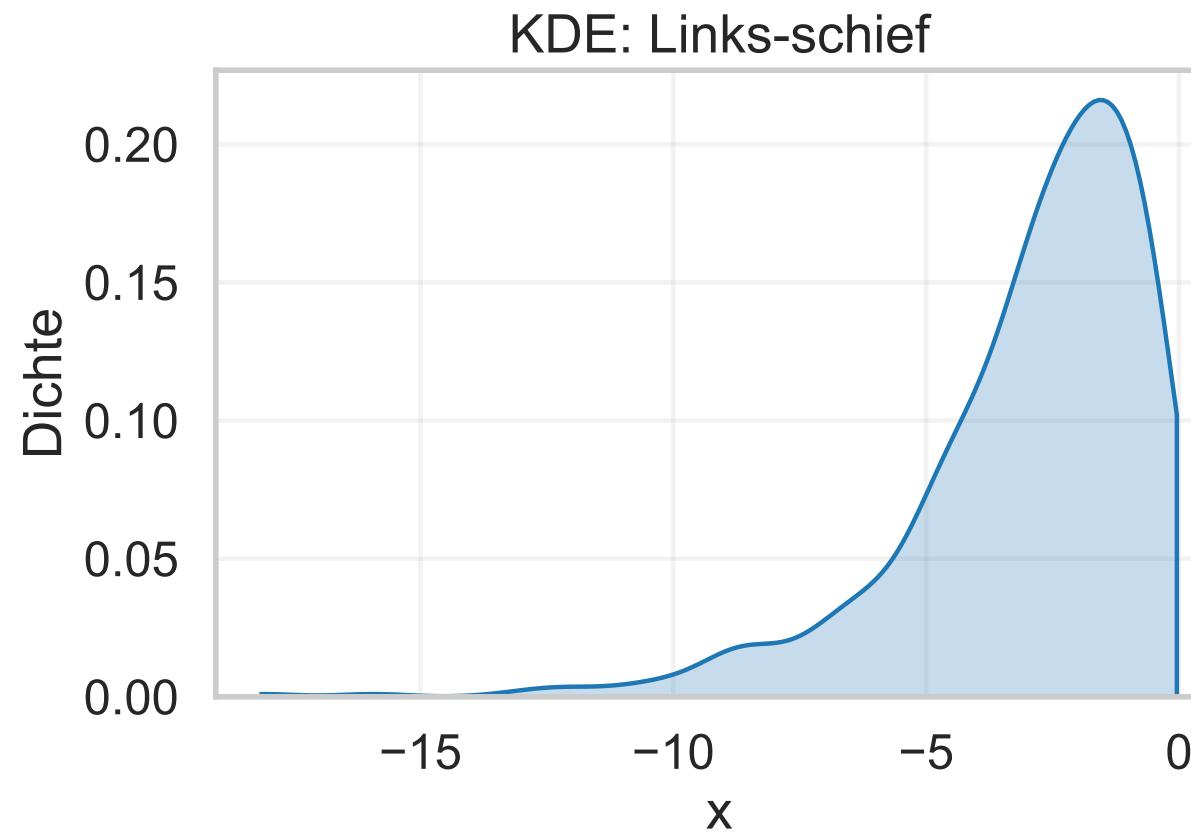
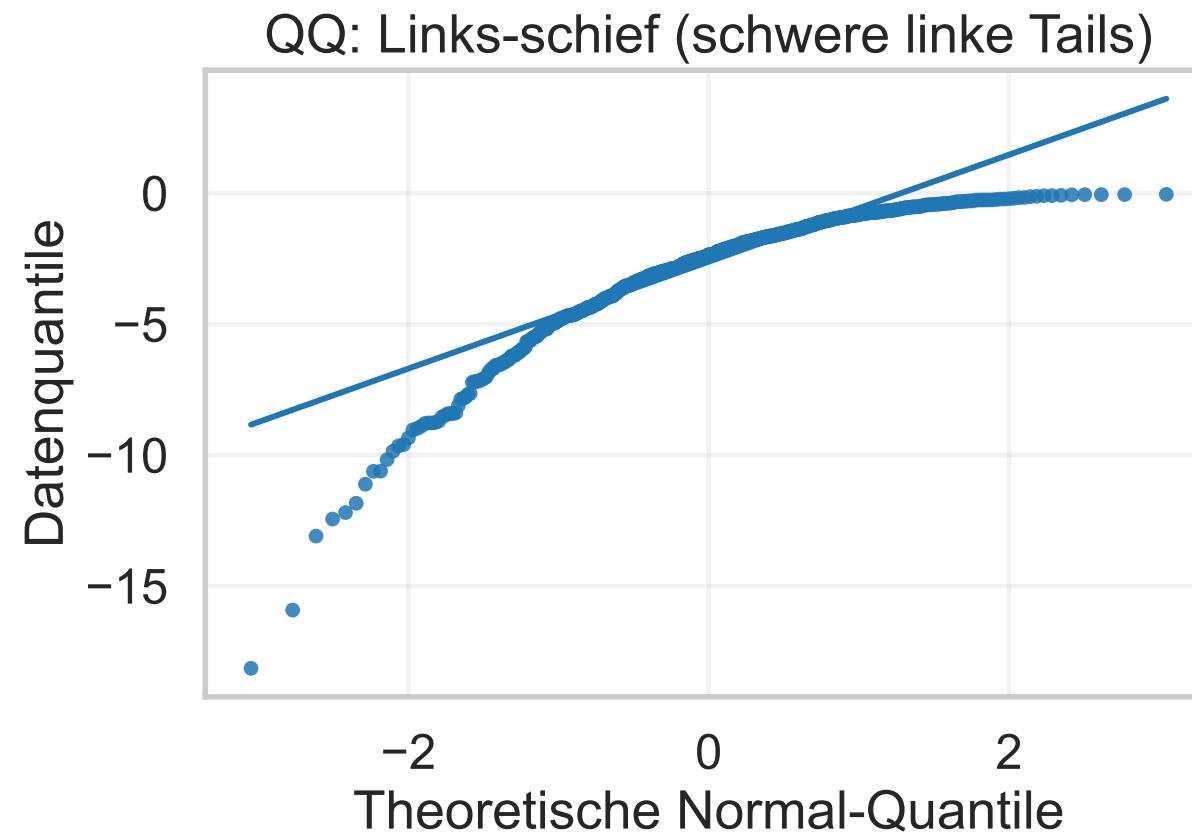
- S-Kurve → Schiefe
- Enden über der Linie → schwere rechte Tails
- Enden unter der Linie → schwere linke Tails
- Immer mit Histogramm oder ECDF abgleichen

Mini-Check: Zeichen für schwere rechte Tails

Normal-QQ – Lesen: Symmetrisch vs. Schiefe und Tails



Schiefe Verteilungen: QQ vs. KDE



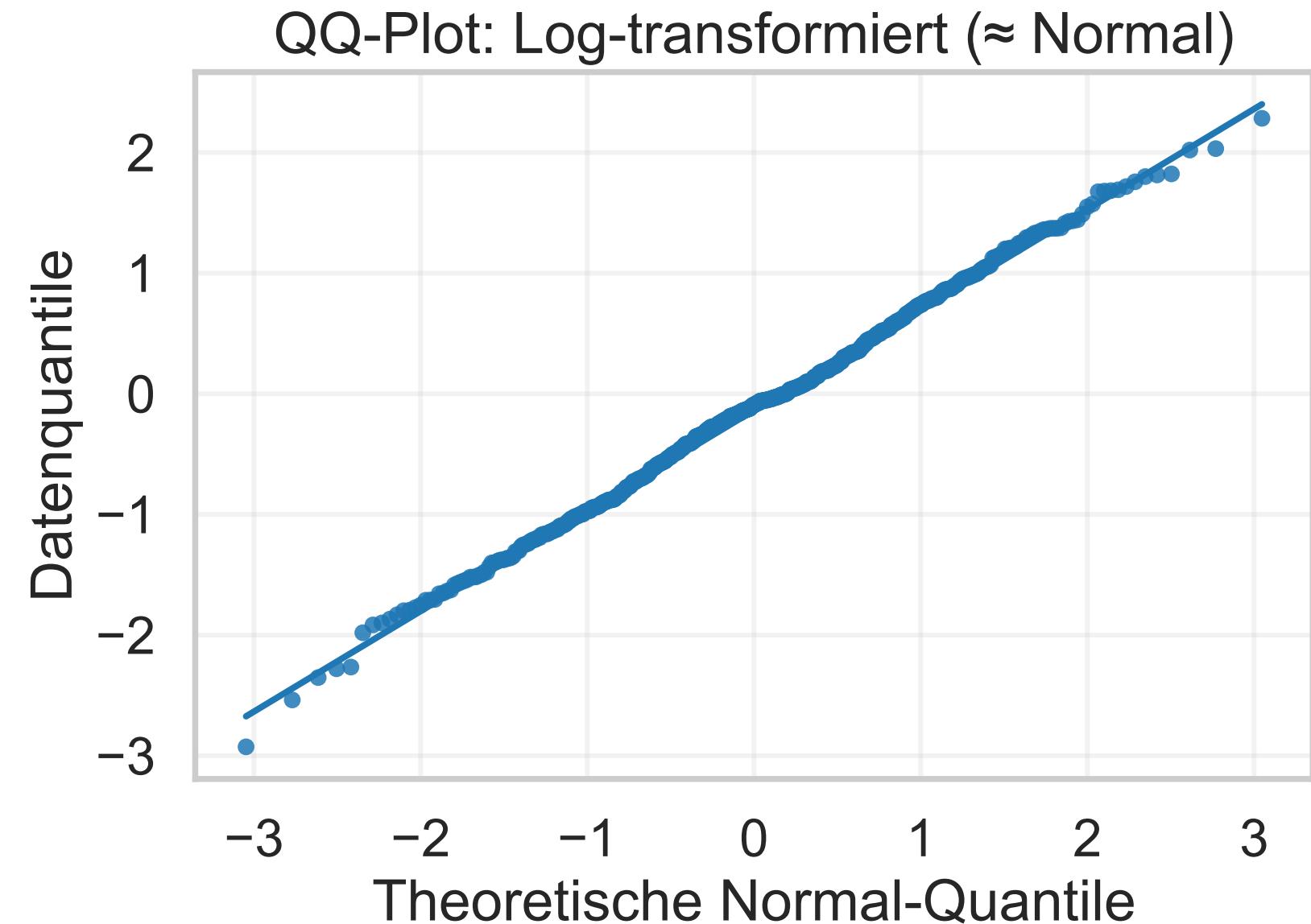
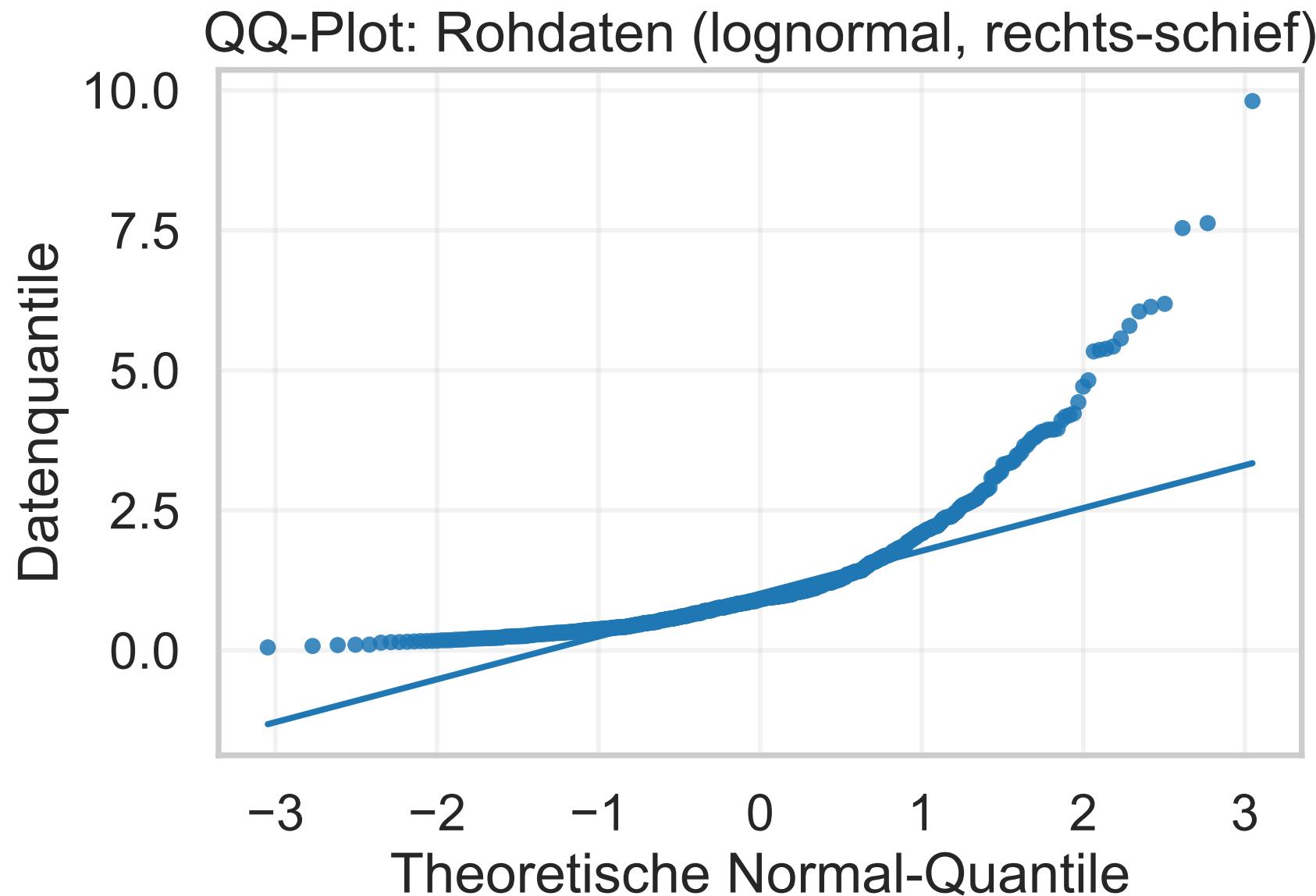
Transformationen & QQ

Log und Wurzel können Normalität nähern.

- Log: für positive, rechts-schiefe Daten
- Wurzel: für Zähldaten (z. B. Ereignisse)
- QQ vor und nach Transformation vergleichen
- Achseninterpretation beachten

Mini-Check: Warum Log bei Null problematisch?

Transformationen & QQ – Log



Python QQ Plot

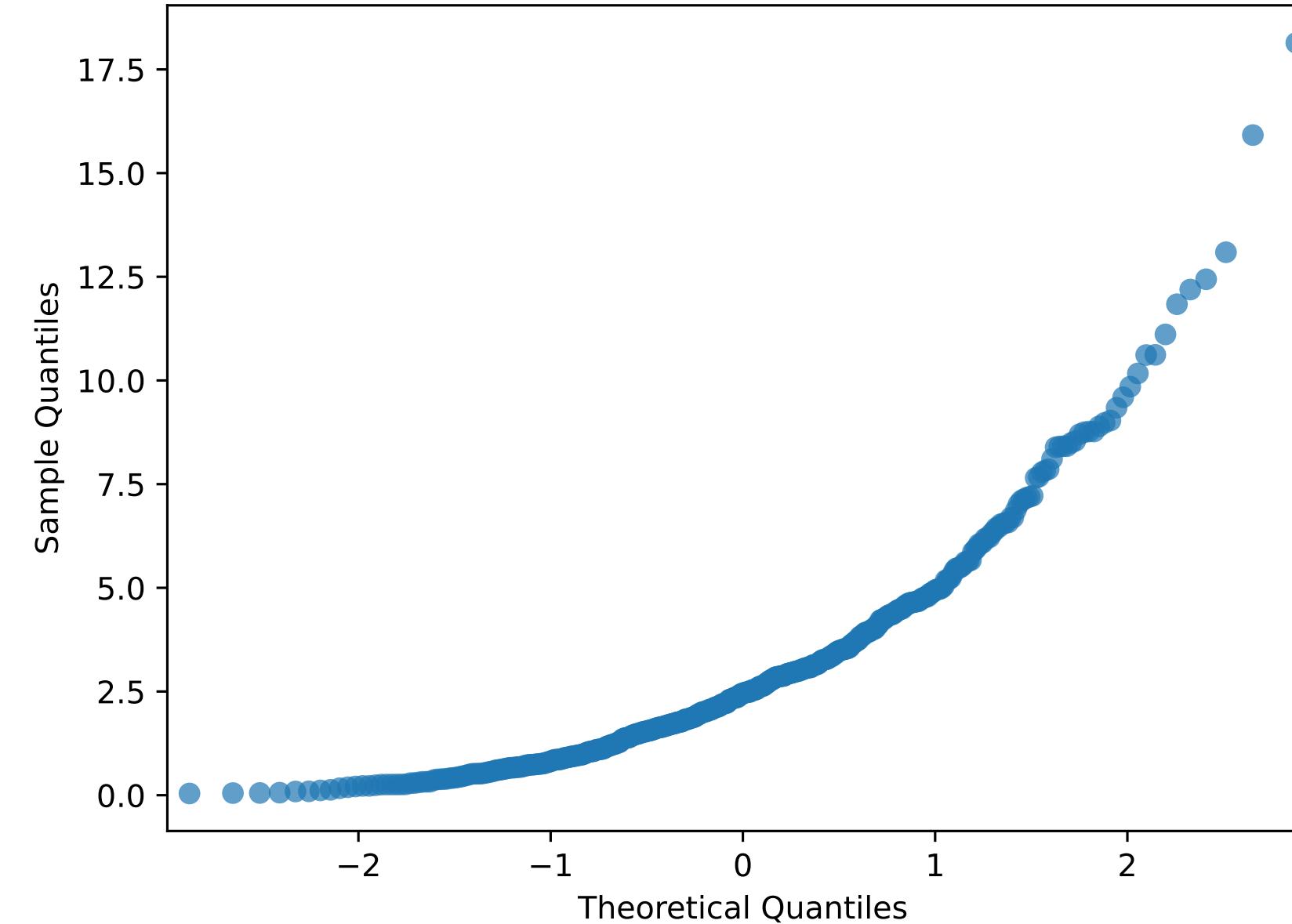
Statsmodels oder SciPy in einer Zeile.

```
import statsmodels.api as sm
sm.qqplot(s, line="45")
# oder
from scipy import stats
stats.probplot(s, dist="norm", plot=plt)
```

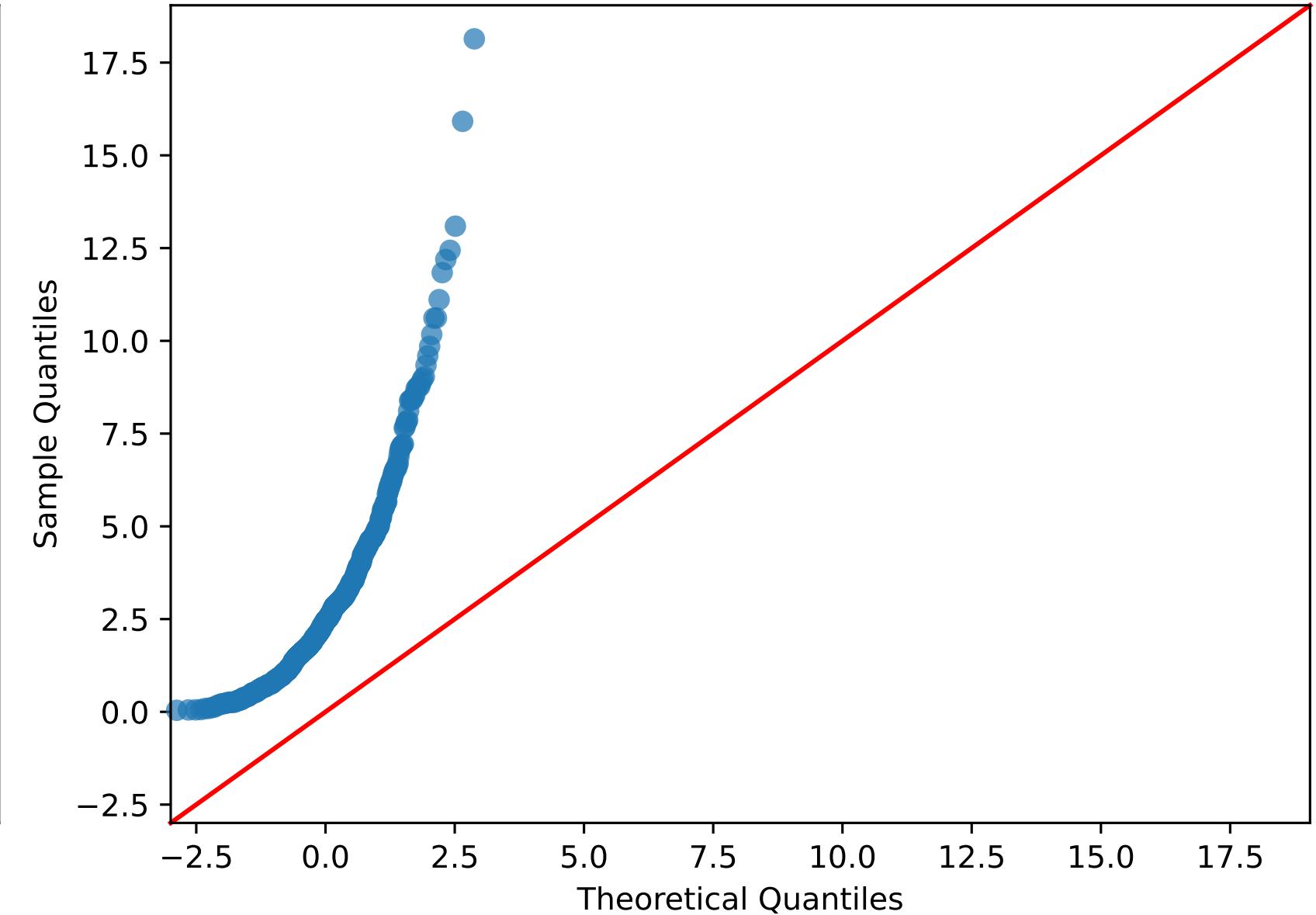
Mini-Check: Bedeutung von line="45"?

Statsmodels QQ-Plot: line=None vs. line='45'

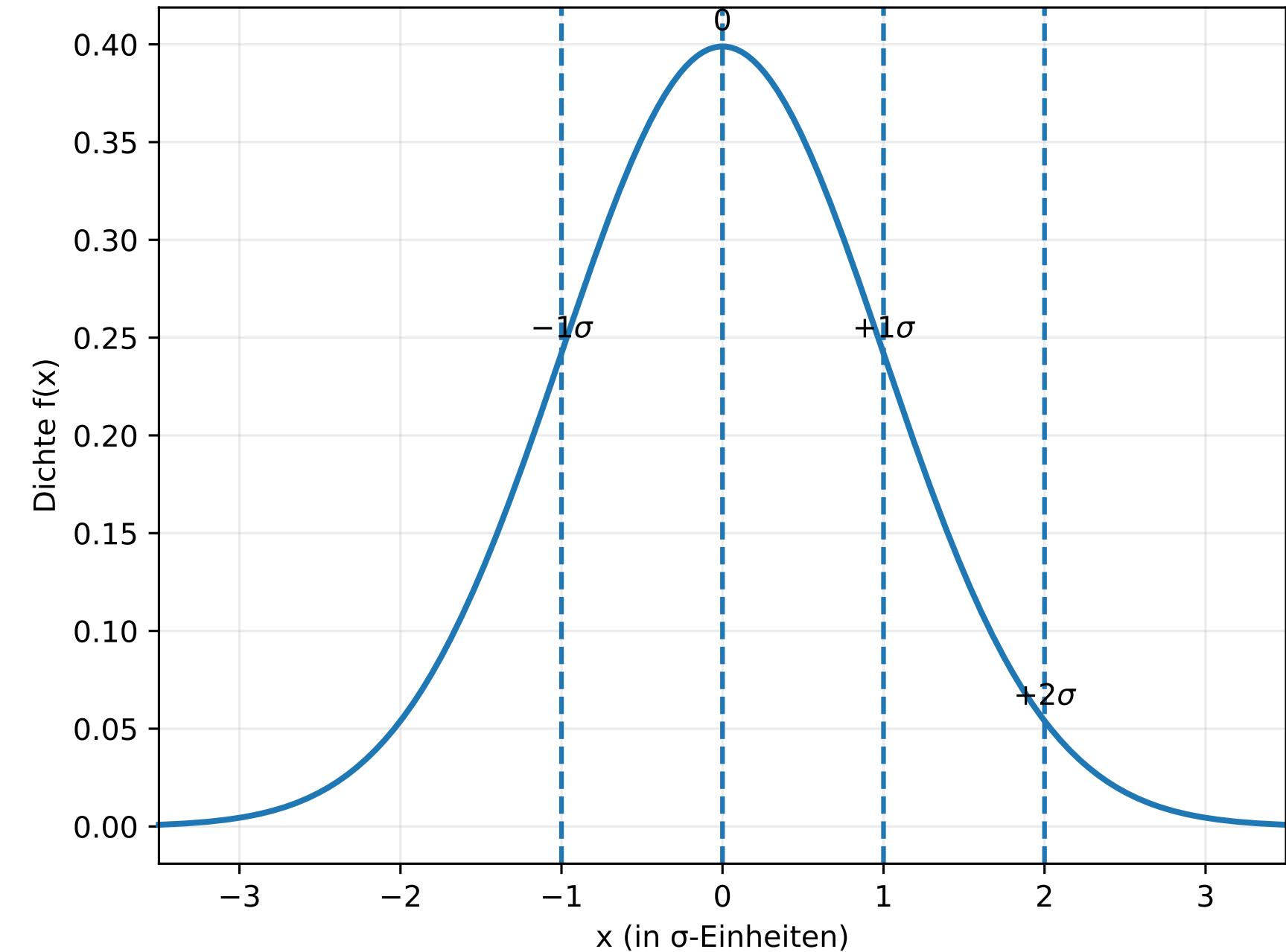
QQ-Plot ohne Referenzlinie



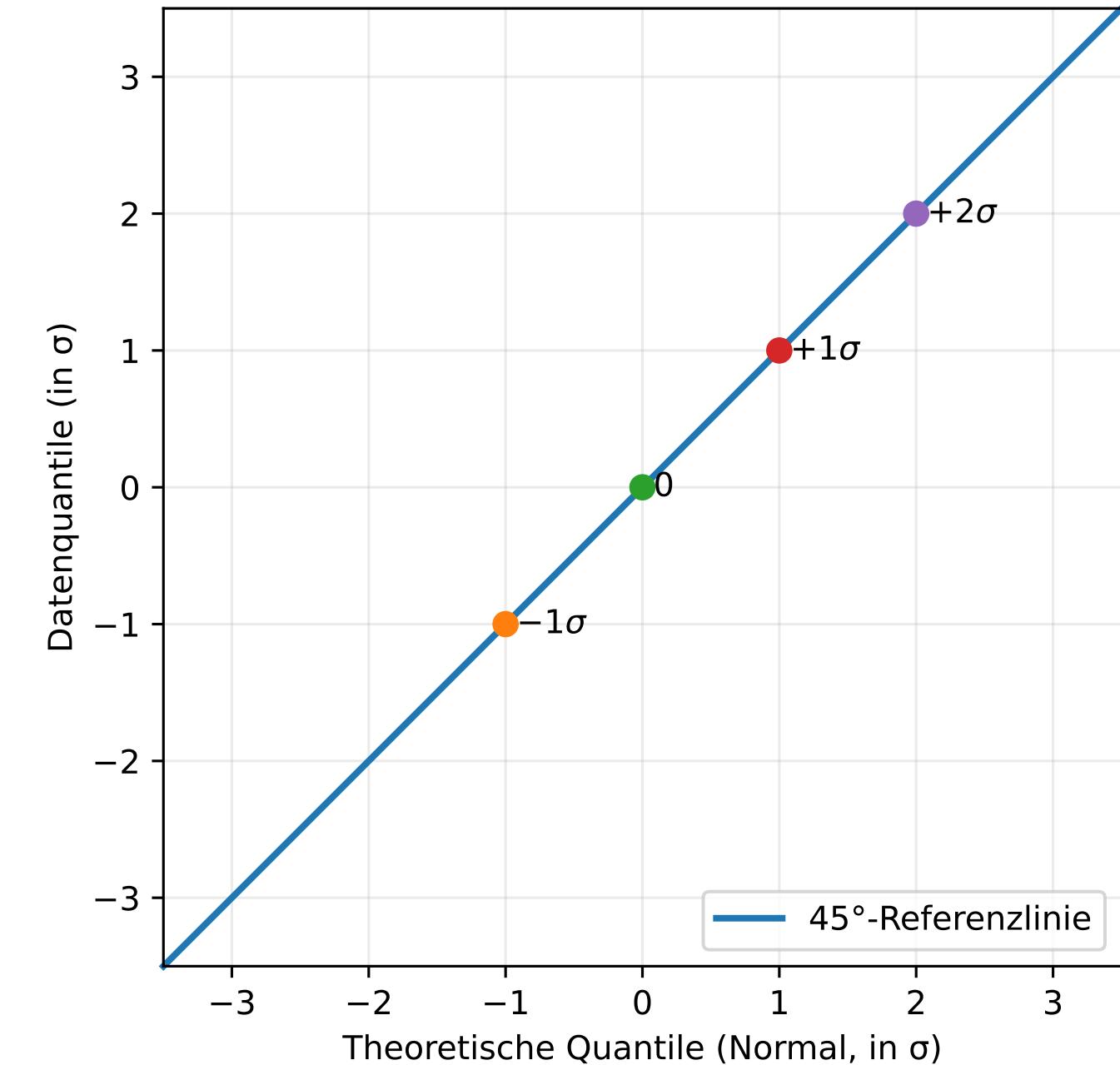
QQ-Plot mit 45° -Linie



Standard-Normalverteilung (PDF) mit σ -Markierungen



Normal-QQ: Theoretische vs. Daten-Quantile



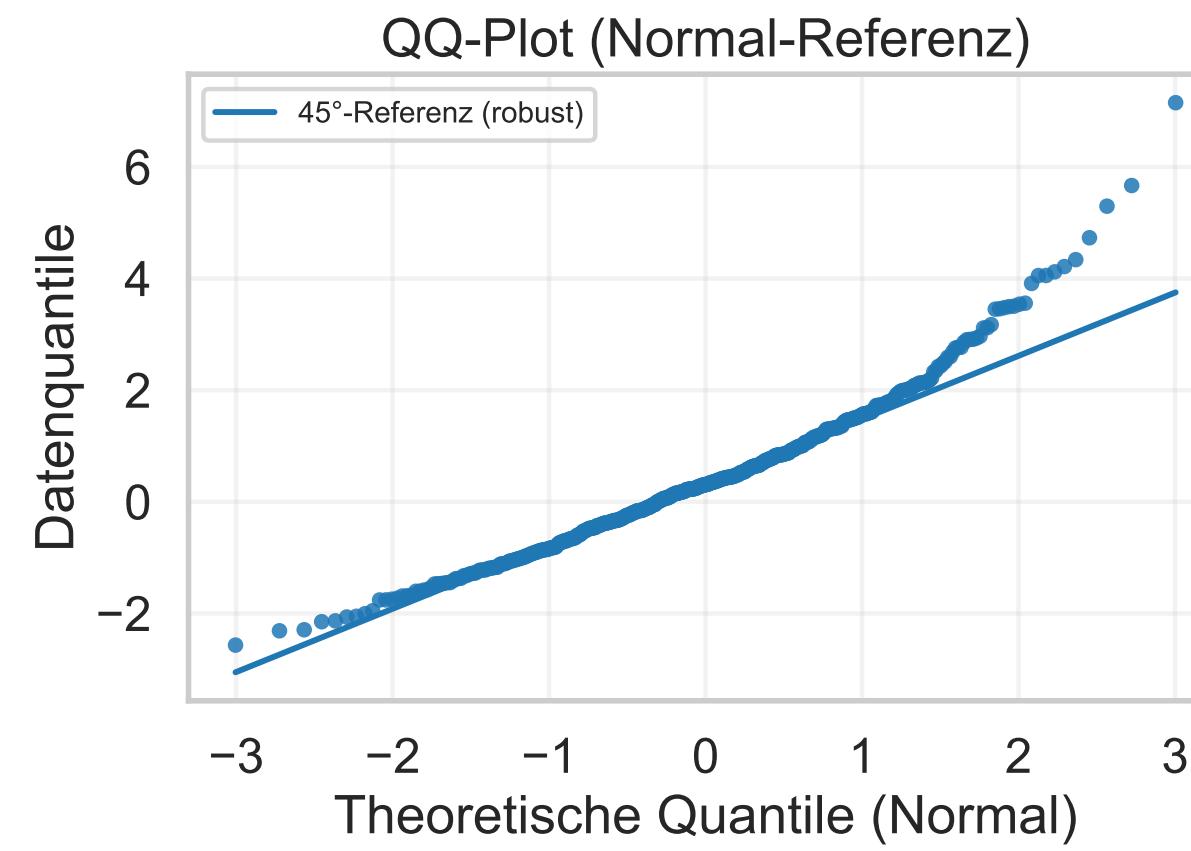
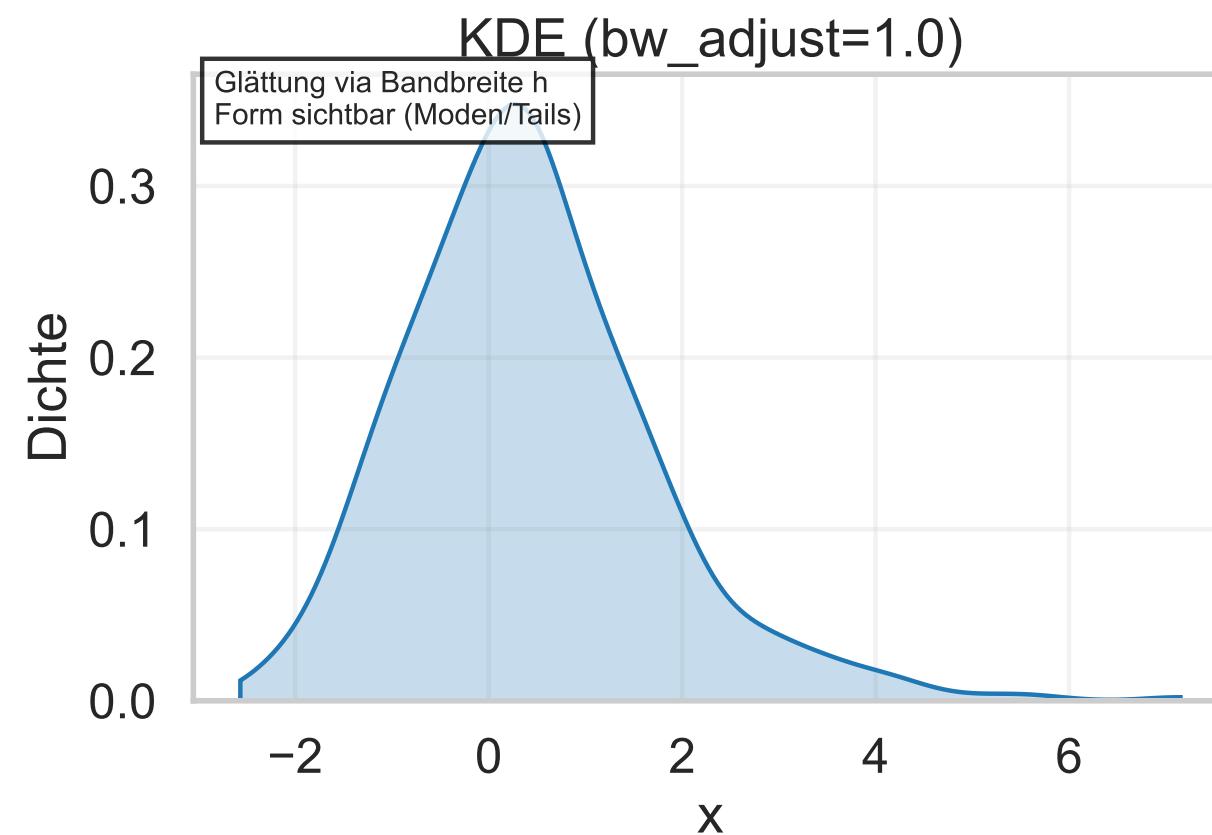
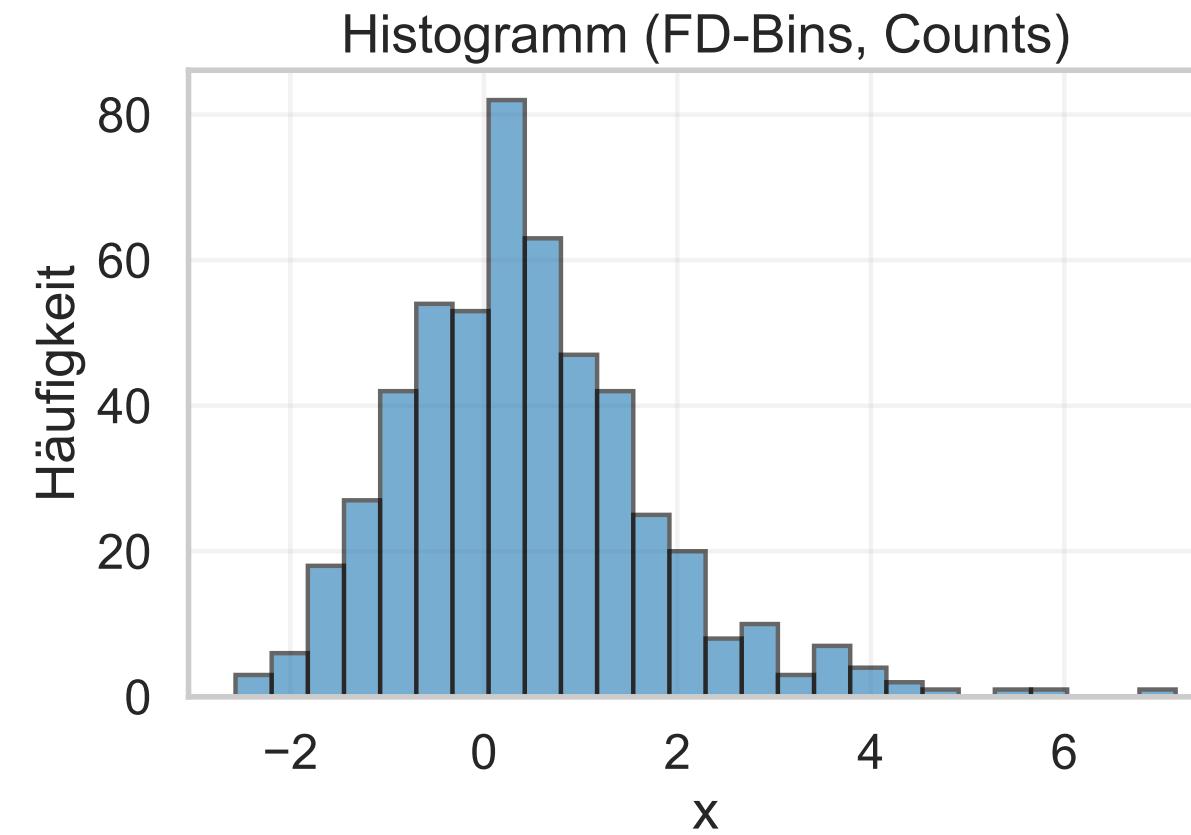
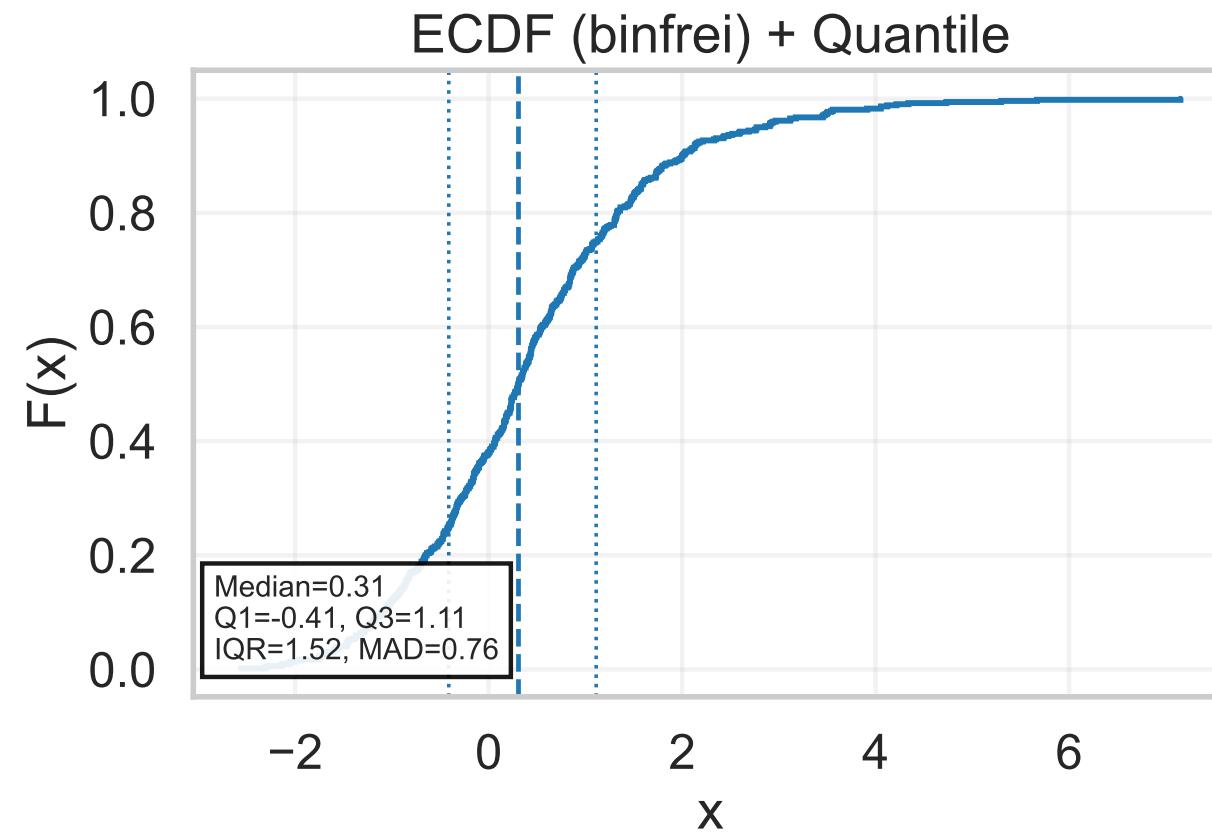
Kombinierte Diagnostik

ECDF, Hist, KDE und QQ gemeinsam nutzen.

- Reihenfolge: ECDF und Kennzahlen, dann Hist+KDE, dann QQ
 - Konsistenz prüfen
 - Entscheidung dokumentieren
 - Kurz und verständlich berichten
- 👉 Mehrere Methoden gemeinsam nutzen = robuste Evidenz.

Key Takeaways – ECDF & QQ

- **ECDF:** binfrei, vollständig, zeigt Quantile direkt
 - **Quantile lesen:** Median = $F(x)=0.5$, Q1/Q3 direkt ablesbar
 - **ECDF-Gruppen:** robust vergleichen, auch bei kleinem n
 - **QQ-Plot:** präzise Formdiagnose im Vergleich zu Referenz (oft Normal)
 - **Muster im QQ:** S-Kurve = Schiefe, Enden abweichend = schwere Tails
 - **Transformationen:** Log/Wurzel können Normalität nähern
 - **Praxis:** Immer mehrere Plots kombinieren für robuste Evidenz
- 👉 ECDF = Überblick • QQ = Feindiagnose • Kombination = Best Practice



Zusammenfassung & Ausblick

Key Takeaways: Vorlesung 3

Thema	Kernpunkte
Outlier	(1) Tukey (IQR) robust (2) mod. Z (Median+MAD) sehr robust (3) Dokumentation nötig
Hist + KDE	(1) Hist = Struktur (2) KDE = Glättung (3) FD-Bins Default (4) Overlay = Best Practice
Box vs. Violin	(1) Box = robuste Kennzahlen (2) Violin = Form/Moden (3) Klein n → Box + Punkte
ECDF & QQ	(1) ECDF binfrei, zeigt Quantile direkt (2) QQ: Gerade Normal, Abweichungen = Schiefe/Tails
Workflow	(1) Robust messen (2) Plots vergleichen (3) Regel anwenden (4) kurz berichten

Quiz - Aktive Wiederholung

Kahoot Quiz VL3

Ausblick: Nächste Woche

Untersuchung von **Zusammenhängen** zwischen zwei Variablen.

- **Masse:** Kovarianz, Pearsons r , Spearman's ρ und Kendalls τ
- **Wichtige Fallen:** Das Simpson-Paradox und die Unterscheidung: **Korrelation \neq Kausalität**

Vorbereitung:

- Practical Statistics for Data Scientists (**PSDS**) Kapitel 2
- Statistics for Data Scientists (**SDS**) Kapitel 3