Befehl	Beschreibung	Gruppe
set.seed(n)	Legt den Startwert für den Zufallsgenerator fest, damit Ergebnisse **reproduzierbar** sind.	
a **%/%** b	quotient	
a **%%** b	remainer	
a**^**b	a to the power of b	Objects, Assignment, and Basic Arithmetic
floor(x)	rundet das Ergebnis auf die nächstkleinere ganze Zahl ab.	
rm()	delete b	
mean()	mean / Durchschnitt	
mean(x%%3==0 | x%%5==0)	Berechnet den Anteil (relative Häufigkeit) der Elemente in x, die durch **3 oder 5 teilbar** sind da TRUE als 1 und FALSE als 0 gemittelt wird.	
mean(A & C)	berechnet den Anteil der Elemente, bei denen sowohl A als auch C TRUE sind — also die gemeinsame Wahrscheinlichkeit (Schnittmenge) beider Bedingungen.	Vectors and Vectorized Operations
var()	variation	
sd()	standard deviation / Standartabweichung	
iseven <- x %% 2 **==** 0	logical Vector	
identical(x, y**)**	Prüft, ob zwei Objekte **vollständig identisch** sind (Werte, Typen, Länge etc.).	
list(poly = poly, stats = stats, is_even = iseven**)**	list of contents	
y**[2:4]**	Select elements 2 through 4 of a vector	
y**[-1]**	Exclude element 1 of a vector	Subsetting and Recycling
y **+** c(1, 2)	Add the vector c(1,2) to y (gets repeated to the number of digits needed for sum)	
factor(other**)**	converts vector in factors (same elements get the same number)	
table(other**)**	frequency table	Factors and Tables
prop.table(other**)**	proportion table	
matrix(1:9, nrow = 3, byrow = TRUE**)**	Creates a 3×3 matrix M filled by rows with the numbers 1 to 9.	
t(M**)**	gegenmatrix	
M **%*%** t(M)	matrix multiplication	
cbind(o1,o2)	fügt die Objekte o1 und o2 spaltenweise zu einer Matrix oder einem Data Frame zusammen.	Matrices and Matrix Algebra
rowSums(M)	**berechnet die Summe der Werte jeder Zeile** der Matrix oder des Data Frames M.	
cbind()	fügt mehrere Vektoren, Matrizen oder Data Frames spaltenweise zusammen und erstellt daraus eine Matrix oder einen Data Frame.	
M**[, 2]**	Extract the second column of matrix	
M[Row,Column]	Select one entry	
rnorm(5**)**	5 normalverteilte zufällige Zahlen	
dnorm(x, mean, sd**)**	Liefert die theoretische **Dichtefunktion** der Normalverteilung N(mean, sd).	
rep("MS", 5**)**	string 5 mal in vector speichern	
data.frame(x = x, initials = initials**)**	create a data frame (normale Tabelle) data.frame() requires columns (vectors) of the same length; matrices with different dimensions cannot be combined directly without reshaping.	Lists and Data Frames
DF**$**x	Select elements from data frame (selects x from DF)	
read.csv("path/to/your/file.csv", sep = ";"**)**	Reading a CSV File with a Different Delimiter (standard is ",")	
write.csv(df, "D:\\Bobdata.csv", row.names=FALSE**)**	Export a Data Frame to a CSV File	
fp <- **tempfile(**fileext = ".csv"**)**	create a temporary file to write the contents to	
plot(DF2$x, DF2$y, pch = 19, main = "Scatter with LS line", xlab = "x", ylab = "y"**)**	create a simple plot with descriptions	Reading/Writing and Basic Plotting
hist(x, breaks, freq, main**)**	histogram	
abline(lm(y **~** x, **data =** DF2**))**	fitted least-squares line	
curve(expr, add = TRUE**)**	Zeichnet eine Funktion (z. B. Dichtekurve dnorm) in ein bestehendes Diagramm.	
z_score <- function(x) { CODE return RESULT **}**	funktion mit x als attribut	Control Flow and Functions
apply()	Wendet eine Funktion auf Zeilen oder Spalten einer Matrix oder eines Dataframes an.	
lapply()	Wendet eine Funktion auf jedes Listenelement an und gibt eine **Liste** zurück.	
sapply()	Wie lapply(), versucht aber das Ergebnis in einen **Vektor oder eine Matrix** zu vereinfachen.	
vapply()	Wie sapply(), aber mit **vordefiniertem Rückgabetyp** (sicherer).	
tapply()	Wendet eine Funktion auf Gruppen eines Vektors an, die durch einen Faktor definiert sind.	The apply Family
mapply()	Wendet eine Funktion auf mehrere Vektoren oder Listen gleichzeitig an (mehrere Argumente).	
rapply()	Rekursive Variante von lapply() für **verschachtelte Listen**.	
eapply()	Wendet eine Funktion auf alle Objekte in einem **Environment** an.	
fit <- lm(y ~ x)	Erstellt ein lineares Regressionsmodell (y abhängig von x).	
summary(fit)	Zeigt detaillierte Modellzusammenfassung (Koeffizienten, R^2, Standardfehler etc.).	Simple Linear Models and Diagnostics
p <- c("1" = 1/2, "2" = 1/3, "3" = 1/6)	setzt die Wahrscheinlichkeiten für die Ereignisse {1}, {2} und {3}.	
P_12 = sum(p[c("1","2")])	wählt die Wahrscheinlichkeiten für die Ereignisse {1} und {2} aus. | Ergibt also c(1/2, 1/3).	
sample(1:100,N, replace = TRUE)	wählt zufällig **N verschiedene Zahlen** aus dem Bereich **1 bis 100** aus	
sample(c("H","T"), 3*N, TRUE)	**erzeugt eine Zufallsauswahl aus "H" und "T" (z. B. Kopf oder Zahl)** mit **insgesamt 3 × N Ziehungen**, **mit Zurücklegen**.	
choose(deck,cards)	die **Anzahl der möglichen Kombinationen**, wenn man cards Karten **ohne Zurücklegen und ohne Reihenfolge** aus einem deck von 52 Karten zieht.	
duplicated(v)	gibt für jedes Element in v TRUE zurück, wenn es schon früher im Vektor vorkam, also ein Duplikat ist, sonst FALSE.	
any()	prüft, ob mindestens ein Element in einem logischen Vektor TRUE ist, und gibt entsprechend TRUE oder FALSE zurück.	
replicate(n, expr)	**führt den Ausdruck expr n-mal aus** und **speichert die Ergebnisse** (meist in einem Vektor oder einer Matrix)	
which(pm > 0.5)[1]	**liefert den Index des ersten Elements** im Vektor pm, das **größer als 0.5** ist.	
dhyper(x, m=8, n=12, k=5)	berechnet die **Wahrscheinlichkeitsdichte der hypergeometrischen Verteilung** – also die **Wahrscheinlichkeit, genau k Erfolge** zu ziehen, wenn man **5 Objekte ohne Zurücklegen** aus einer **Gesamtmenge von 20 (8 Erfolge, 12 Misserfolge)** zieht.	
length()	gibt **die Anzahl der Elemente** eines Vektors, einer Liste oder eines ähnlichen Objekts zurück.	
sort()	**ordnet die Elemente eines Vektors standardmäßig aufsteigend** (optional auch absteigend).	
tail(pmf_hat, 1)	gibt **das letzte Element** des Vektors oder Data Frames pmf_hat zurück.	
rgeom(n, prob = p)	**erzeugt n Zufallswerte aus der geometrischen Verteilung** mit **Erfolgswahrscheinlichkeit p** (Anzahl der Fehlversuche vor dem ersten Erfolg).	
seq(0,1,0.01)	**erzeugt eine Zahlenfolge von 0 bis 1 in Schritten von 0,01**.	

seq_along(grid)	**erzeugt eine Zahlenfolge von 1 bis zur Länge von grid**, also die passenden Indizes für grid
rbinom ()	**erzeugt Zufallszahlen aus der Binomialverteilung**, also die Anzahl der Erfolge in einer festen Anzahl von Versuchen mit gegebener Erfolgswahrscheinlichkeit.
max()	**gibt den größten Wert** eines numerischen Vektors oder Objekts zurück (optional unter Ignorieren von NA).
which.max()	**liefert den Index des größten Werts** im Vektor
runif(N)	erzeugt N gleichverteilte Zufallszahlen im Intervall [0,1]
sqrt()	**berechnet die Quadratwurzel** der übergebenen Zahl(en).
ecdf(x)	**erstellt die empirische Verteilungsfunktion** der Daten in x, also eine Funktion, die für jeden Wert den **Anteil der Beobachtungen ≤ diesem Wert** angibt.
pnorm(2)	**gibt die Wahrscheinlichkeit an, dass eine standardnormalverteilte Zufallsvariable kleiner oder gleich 2 ist** (also P(X≤2)P(X \le 2)P(X≤2)).
pnorm(6) - pnorm(2)	Zwischen 2 und 7
hist()	**erstellt ein Histogramm**, das die **Häufigkeitsverteilung** von numerischen Daten grafisch darstellt.
abs()	**berechnet den Absolutbetrag** einer Zahl bzw. eines numerischen Vektors. **(Abstand zur 0)**
rexp(n, lambda)	**erzeugt n Zufallswerte aus der Exponentialverteilung** mit **Rateparameter lambda**.
sweep()	*wendet eine Operation (z. B. +, −, , /) zeilen- oder spaltenweise auf ein Array/Matrix an* , meist zum **Addieren, Subtrahieren, Multiplizieren oder Dividieren mit einem Vektor**.
par(mar = c(4,4,2,1))	**legt die Plot-Ränder in R fest** (unten, links, oben, rechts) in Zeilenbreite.
matplot()	**stellt mehrere Spalten einer Matrix gleichzeitig in einem Diagramm dar**, typischerweise als Linien oder Punkte.
selectDWD()	filtert und wählt passende DWD-Datensätze bzw. Stationen anhand von Kriterien wie Parameter, Zeitraum oder Stations-ID aus.
dataDWD(meta, read = FALSE)	**lädt die zu den Metadaten meta gehörenden DWD-Dateien herunter, ohne sie direkt einzulesen**, und gibt stattdessen **Dateipfade bzw. Dateiinformationen** zurück.
readDWD(file)	**liest eine DWD-Datendatei aus einer lokalen Datei ein** und gibt die enthaltenen Wetterdaten als **Data Frame/Tibble** zurück.
%>%	Pipeline
select(-eor)	**entfernt die Spalte eor aus einem Data Frame** (typischerweise mit dplyr::select).
na.omit()	**entfernt alle Zeilen, die mindestens einen NA-Wert enthalten**, aus einem Objekt (z. B. Data Frame oder Vektor).
as.Date(df$MESS_DATUM)	**wandelt die Spalte MESS_DATUM aus dem Data Frame df in ein Date-Objekt um**.
mutate()	**fügt neue Spalten zu einem Data Frame hinzu oder verändert bestehende Spalten**, meist auf Basis bereits vorhandener Variablen (aus dplyr).
sum()	**berechnet die Summe aller Elemente** eines numerischen Vektors oder Objekts (optional ohne NA).
ggplot(bwt, aes(x = bwt)) + **geom_histogram(binwidth = 100)** + ggtitle("Age with width = 100 and ggplot")	erstellt mit ggplot2 ein Histogramm der Variable bwt, verwendet Klassenbreite 100 und setzt den Titel des Diagramms auf „Age with width = 100 and ggplot“.
ggplot(bwt, aes(x = bwt)) + stat_ecdf()	**erstellt mit ggplot2 die empirische Verteilungsfunktion (ECDF)** der Variable bwt, also den **Anteil der Beobachtungen ≤ einem gegebenen Wert**.
nrow()	**gibt die Anzahl der Zeilen** eines Data Frames oder einer Matrix zurück.
median()	**berechnet den Median**, also den **mittleren Wert** eines numerischen Vektors.
summary(bwt$bwt)	**gibt eine statistische Zusammenfassung** der Variable bwt aus, also **Minimum, 1. Quartil, Median, Mittelwert, 3. Quartil und Maximum**.
quantile(bwt$lwt, 0.75)	**berechnet das 75%-Quantil (3. Quartil)** der Variable lwt
quantile(bwt$lwt, 0.25)	**berechnet das 25%-Quantil (1. Quartil)** der Variable lwt
(quantile(bwt$lwt, 0.75) - quantile(bwt$lwt, 0.25)	Interquartile Range (IQR), also die Spannweite zwischen dem 3. und 1. Quartil.
select()	**wählt gezielt Spalten aus einem Data Frame aus oder schließt bestimmte Spalten aus** (typischerweise aus dem Paket **dplyr**).
filter()	**wählt Zeilen aus einem Data Frame aus**, die eine oder mehrere **logische Bedingungen erfüllen** (typischerweise aus dem Paket **dplyr**).
filter(!is.na(wet))	**entfernt alle Zeilen, in denen die Variable wet den Wert NA hat**, und behält nur vollständige Beobachtungen für wet.
case_when()	**erstellt neue Variablen anhand mehrerer Wenn-Dann-Bedingungen** und weist je nach erfüllter Bedingung entsprechende Werte zu (aus **dplyr**).
as_tibble()	**wandelt ein Objekt (z. B. Data Frame oder Matrix) in ein Tibble um**, eine moderne, übersichtlichere Form eines Data Frames (aus **tibble**).
summarise(across(everything(), ~sum(is.na(.))))	**zählt für jede Spalte eines Data Frames die Anzahl der NA-Werte** und gibt das Ergebnis als eine Zeile zurück.
count(wet)	**zählt, wie oft jeder unterschiedliche Wert in der Variable wet vorkommt**, und gibt die Häufigkeiten aus.
ggplot(aes(x = t_mean, colour = season))+ stat_ecdf(na.rm = TRUE)	**erstellt mit ggplot2 empirische Verteilungsfunktionen (ECDFs)** der Variable t_mean, **getrennt nach season**, und **ignoriert NA-Werte**.
ggplot(aes(x = t_mean, colour = season))+ geom_boxplot(na.rm = TRUE)	**erstellt mit ggplot2 Boxplots der Variable t_mean, getrennt nach season**, und **ignoriert NA-Werte**.
ggplot(aes(x = wet, colour = season))+ geom_bar()	**erstellt mit ggplot2 ein Balkendiagramm**, das die **Häufigkeiten der Werte von wet darstellt**, **farblich getrennt nach season**.
lm()	**passt ein lineares Regressionsmodell an**, um den **Zusammenhang zwischen einer Zielvariable und einer oder mehreren erklärenden Variablen** zu schätzen.
tidy(fit)	**wandelt die Ergebnisse eines Modellobjekts (z. B. aus lm()) in eine übersichtliche Tabellenform** um, mit **Koeffizienten, Standardfehlern, t-Werten und p-Werten** (aus dem Paket **broom**).
glance(fit)	**liefert eine kompakte Zusammenfassung des Modells** (z. B. R^2, adj. R^2, AIC, BIC, Residualstandardfehler) **in einer einzigen Tabellenzeile** (aus **broom**).
augment(fit)	**ergänzt die ursprünglichen Daten um modellbezogene Werte**, z. B. **Vorhersagen, Residuen und Hebelwerte**, und gibt sie als Tabelle zurück (aus **broom**).
ggplot(diag_tbl, aes(x = .fitted, y = .resid)) + geom_point() + geom_hline(yintercept = 0, linetype = "dashed") + labs(title = "Residuals vs Fitted", x = "Fitted values", y = "Residuals")	**erstellt ein Residuen-gegen-Vorhersage-Diagramm**, um **Muster, Heteroskedastizität oder Modellfehler** im linearen Regressionsmodell zu erkennen.