Prüfung

UNIVERSITÄT
BERN

OESCHGER CENTRE
CLIMATE CHANGE RESEARCH

- Montag, 10.02.2025
- 14:15 – 15:15
- Hauptgebäude, Aula 210

- Wer ein Auslandssemester macht, bitte persönlich melden

Daten auswählen aus data.frame bzw. matrix

UNIVERSITÄT BERN
OESCHGER CENTRE CLIMATE CHANGE RESEARCH

Auswahl in kleinerem data.frame speichern

```
> sommer <- saison[saison[,2]=='Sommer(JJA)',]
                # Objekt saison, aber nur Zeilen, wo in 2. Spalte 'Sommer (JJA)' steht
```

Einzelne Variable/Spalte auswählen und in Vektor speichern

```
> sommer_temp_genf <- sommer[,3] oder
> sommer_temp_genf <- sommer[,'Genf_Mitteltemperatur']

> jahre <- sommer[,1]
```

Rechnungen über ganze Spalten / Zeilen mit APPLY

UNIVERSITÄT BERN
OESCHGER CENTRE CLIMATE CHANGE RESEARCH

> *apply(saison[,3:6], **2**, mean)* ...**2** = *spaltenweise (1 wäre zeilenweise)*

Genf_Mitteltemperatur	Genf_Niederschlagssumme	GrStBernhard_Mitteltemperatur	GrStBernhard_Niederschlagssumme
9.732662	234.364206	-1.182476	549.379418

Nur für Auswahl (z.B. Sommersaison; siehe vorherige Folie)

> *apply(saison[saison[,2]=='Sommer(JJA)',3:6], 2, mean)*

Genf_Mitteltemperatur	Genf_Niederschlagssumme	GrStBernhard_Mitteltemperatur	GrStBernhard_Niederschlagssumme
18.168750	254.375893	5.987202	432.576786

oder direkt mit dem neu erzeugen data.frame 'sommer':

> *apply(sommer[,3:6], 2, mean)*

oder einfach das Mittel des vorher erzeugten Vektors 'sommer_temp_genf':

> *mean(sommer_temp_genf)*

Rechnungen nach Klassen

UNIVERSITÄT BERN
OESCHGER CENTRE CLIMATE CHANGE RESEARCH

Funktion ***aggregate()***

Syntax: *aggregate(numerischer Vektor(en), list(Klassen), FUN= Funktion)*

```
> s.agg <- aggregate(saison[,3:6], list(saison[,2]), FUN = mean)
```

	Group.1	Genf_Mitteltemperatur	Genf_Niederschlagssumme	GrStBernhard_Mitteltemperatur
1	Fruehling(MAM)	9.369940	209.3482	-3.325595238
2	Herbst(SON)	9.905706	264.9090	-0.006306306
3	Sommer(JJA)	18.168750	254.3759	5.987202381
4	Winter(DJF)	1.487798	209.0964	-7.374702381

	GrStBernhard_Niederschlagssumme
1	596.0366
2	558.9613
3	432.5768
4	610.0286

Grafiken in R

UNIVERSITÄT BERN
OESCHGER CENTRE CLIMATE CHANGE RESEARCH

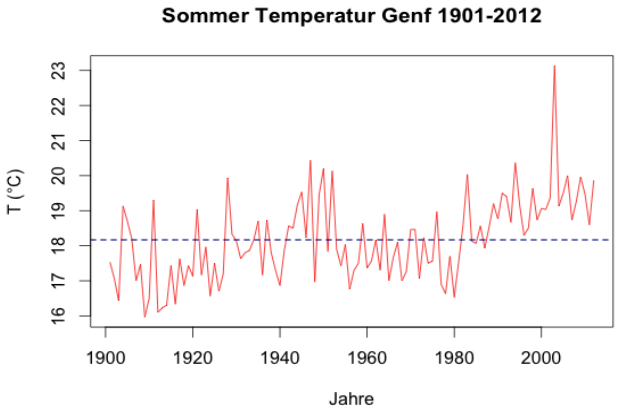
„High level' Plot Funktionen – „zaubern aus allem eine Grafik', jedoch nicht immer nach Wunsch, je nach Daten/Objekt

plot(), hist(), boxplot(), pairs(),...

> *plot(saison)*

> *plot(saison[,'Genf_Mitteltemperatur'])*

> *plot(sommer[,3])*

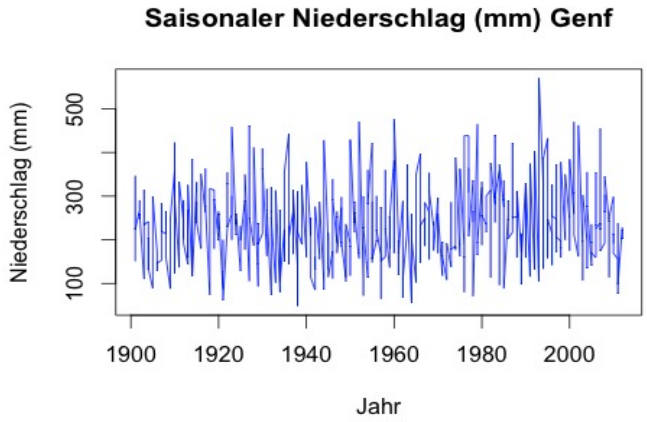


„Low level' Plot Funktionen – kann man einer bestehenden Grafik nur hinzufügen

lines(), abline(), points(), ...

> *abline(h=18)*

Funktion *plot(x,y)*



> *?plot*

> *plot(sommer[,1],sommer[,3])*

> *plot(sommer[,1],sommer[,4],*

type ='l', ...Darstellung: Punkte „p', Linie „l',

xlab='Jahr', ...Beschriftung x-Achse,

ylab='Niederschlag (mm)', ...Beschriftung y-Achse,

main='Saisonaler Niederschlag (mm) Genf', ...Überschrift

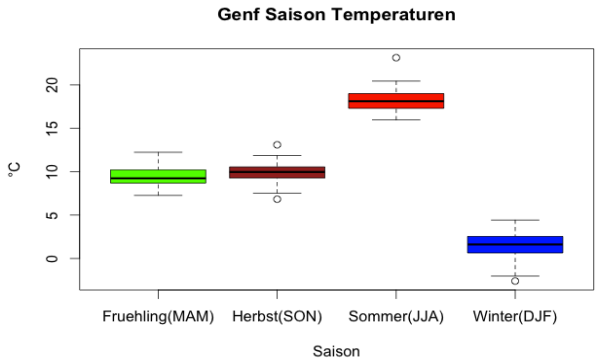
col='blue', ...Farbe z.B. „green', 'red', ...

ylim=c(80,600), xlim=c(...,...),

...)

weitere Grafik-Parameter findet ihr mit: *?par*

Funktion *boxplot(y~x)*



> *?boxplot*

> *boxplot(saison[,3] ~ saison[,'Saison'])* … *boxplot(daten~gruppeninfo)*

> *boxplot(saison[,3] ~ saison[,2],*

main='Genf Saison Temperaturen', ... Überschrift

ylab='° C', , ... Beschriftung y-Achse,

xlab='Saison', , ... Beschriftung x-Achse,

col=c('green', 'brown', 'red','blue') ... Farbe(n)

...)

Funktion *hist(x)*

UNIVERSITÄT BERN
OESCHGER CENTRE
CLIMATE CHANGE RESEARCH

```
> ?hist
> hist(sommer[,3])
```

```
> hist(sommer[,3],
        main='Histogramm',
        ylab='Häufigkeiten',
        xlab='°C',          ,
        col='green',
        breaks=20,
        ...)
```

... Überschrift
... Beschriftung y-Achse,
... Beschriftung x-Achse,
... Farbe(n)
... ungefähre Anzahl Klassen

Funktion *barplot(x)*

UNIVERSITÄT BERN

OESCHGER CENTRE
CLIMATE CHANGE RESEARCH

```
> ?barplot
> s.agg <- aggregate(saison[,3:6], list(saison[,2]), FUN = mean)
```

… von Folie mit aggregate() Funktion

```
> barplot(s.agg[,2])
```

```
> barplot(s.agg[,3],
        names.arg=s.agg[,1],                    ... Beschriftung der bars
         main='Barplot',                        ... Überschrift
         ylab='°  C',                           ... Beschriftung y-Achse,
         xlab='Saison',      ,                  ... Beschriftung x-Achse,
         col=c('green', 'brown', 'red','blue') ,  ... Farbe(n)
         ...)
```

Hinzufügen von „low-level' Plot Funktionen

UNIVERSITÄT BERN
OESCHGER CENTRE CLIMATE CHANGE RESEARCH

zuerst High-Level Funktion z.B. ***plot(x,y),*** Grafik öffnet sich

> *plot(sommer[,1], sommer[,3],type='l',...)*

Hinzufügen von „low-level' Funktionen bei offener Grafik

> *abline(h=20)*	... **h**orizontale Linie bei 20 (auch **v**ertikal)
> *abline(h=mean(sommer[,3]),col='blue')*	... horiz. Linie bei Mittelwert
> *points(sommer[,1], sommer[,3],*	... Punkte hinzufügen
pch=2)	... Punkt-typ
> *lines(**x,y**,...)*	
> *legend(...)*	... Legende hinzufügen

Grafikfunktion *par (...)* vor plots

UNIVERSITÄT BERN
OESCHGER CENTRE CLIMATE CHANGE RESEARCH

> *?par* ... Grafikparameter für alle Grafiken festlegen

> *par(mfrow=c(2,1))* ... plot mit 2 Grafiken untereinander (2 Zeilen)

> *barplot(s.agg[,2] names.arg=s.agg[,1])*

> *plot(sommer[,1], sommer[,3],type='l',xlab='Jahr',col=„red')*

Legende hinzufügen, nach plots (low-level Funktion), *?legend*

> *legend('topleft',* ... wo soll sie plaziert werden

legend=`Sommer', ... Beschriftung des/der Labels

col='red' , ... Farbe(n)

lty=1, ... wenn Linie->Linientyp (1=durchgezogen)

wenn Symbol: pch (pointcharakter)=(z.B.) 1

cex=0.8, ...) ... Grösse der Legende (relative zu 1)

UNIVERSITÄT
BERN

OESCHGER CENTRE
CLIMATE CHANGE RESEARCH

ÜBUNGEN 2

R-Übungen 2

ACHTUNG: immer Teil des Codes einzeln ausführen, wenn etwas nicht funktioniert! Hier z.B.:
saison[,2]=='Fruehling(MAM)'

UNIVERSITÄT BERN
OESCHGER CENTRE CLIMATE CHANGE RESEARCH

2.1) Grafik erstellen

> Extrahiert aus den saisonalen Daten erstens nur die Frühlingsdaten, zweitens nur die Sommerdaten und drittens nur die Herbstdaten z.B.

fruehling <- saison[saison[,2]=='Fruehling(MAM)',]

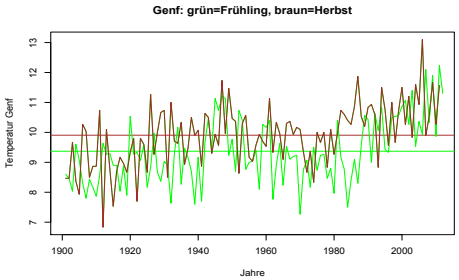
> Erstellt einen Plot, mit den Jahren auf der x-Achse und der Temperatur in Genf auf der y-Achse. Stellt dabei die **Frühlings-, Sommer- und Herbst**temperaturen als Linien mit unterschiedlichen Farben im gleichen Plot dar:

> zuerst *plot(x,y,col=' ', xlab=' ',...)*

> dann mit *lines(x,y,col=..)* weitere Saisons

> Vergebt eine Überschrift und beschriftet auch beide Achsen

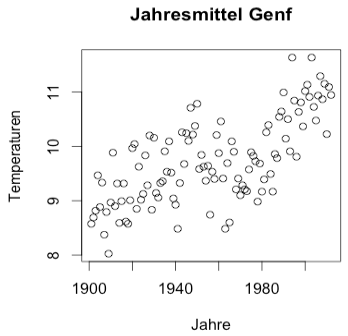
> Fügt Linien der beiden Mittelwerte hinzu mit *abline(h=...)*



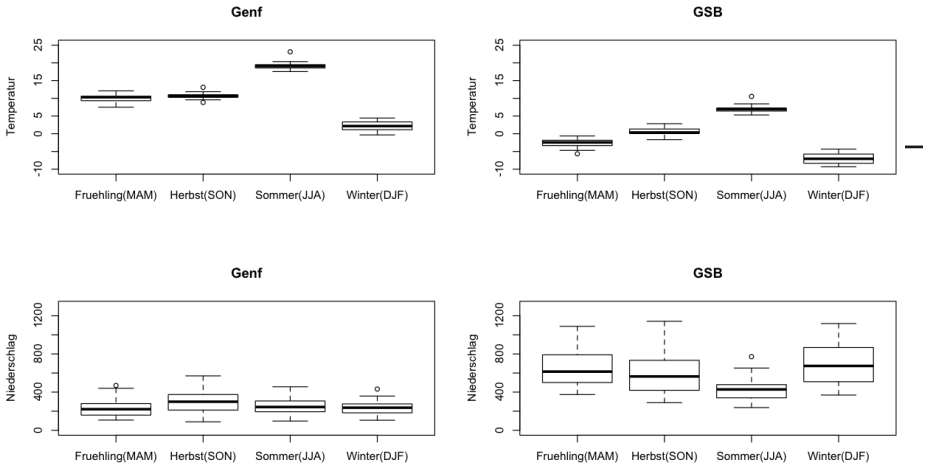
2.2)

> Erstellt mittels *aggregate()* die **Jahresmittelwerte** der Temperatur für Genf und stellt diese in einem Scatterplot mit Punkten *plot(x,y)* dar.

> Beschriftet die Achsen und vergebt einen Titel.



R-Übungen 2



2.3)

> Wählt den **Zeitraum 1981-2010,** z.B. *zeit <- saison[,1]>=1981 & …*

> Stellt die Temperatur- und Niederschlagsverteilungen der Saisons in Genf und Gr. S. Bernhard für diesen Zeitraum in vier *boxplot()* dar.

> Das Grafikausgabefenster kann mit *par(mfrow=c(2,2)* in 2 Zeilen und 2 Spalten geteilt werden.

> Beschriftet wieder die Achsen und vergebt Titel. Achtet darauf, für beide Stationen gleiche y-Achsen zu wählen, so dass die Plots gut visuell vergleichbar sind, z.B. bei Niederschlag je *ylim=c(0,1300)*

> Das Grafikausgabefenster mit *par(mfrow=c(1,1) wieder auf 1* Zeilen und 1 Spalten zurücksetzen.

b
UNIVERSITÄT
BERN

OESCHGER CENTRE
CLIMATE CHANGE RESEARCH

Graphik speichern

UNIVERSITÄT BERN
OESCHGER CENTRE CLIMATE CHANGE RESEARCH

als eps, jpg, pdf,... 'plot.pdf'

in R-Studio:

'Export' … 'Save Plot as PDF'oder 'Image' … (bei 'Image' gewünschtes Format wählen)... 'Directory' angeben (=Pfad) oder manuell

> *pdf('name.pdf', width=9, height=4.5, …)* ... **oder**

> png(…) … zum Grafikdatei anlegen/öffnen, je nach Formatwunsch

> *plot(…)* *… Grafik erstellen*

> *dev.off()* *… Grafik abschliessen*

Daten speichern

UNIVERSITÄT BERN
OESCHGER CENTRE CLIMATE CHANGE RESEARCH

als ASCII Textdatei 'file.txt'

> *write.table(saison,* … welches Objekt soll gespeichert werden

file='/…/…/…/file.txt') … Pfad und gewünschten Dateinamen angeben

Es gibt spezielle Funktionen zum Lesen und Schreiben vieler weiterer Datenformate. Diese findet ihr bei Bedarf leicht im Internet.

Umgang mit Fehlwerten in R

UNIVERSITÄT BERN
OESCHGER CENTRE CLIMATE CHANGE RESEARCH

R kodiert Fehlwerte mit 'NA'

bei Funktionen:

> *a <- c(1, 3, 4, NA, 5)*

> *sum(a)*

[1] NA ...man muss angeben wie mit NAs umgehen will mit **na.rm**

> *sum(a, **na.rm=TRUE**)* ...z.B. na.rm (NA remove) TRUE

[1] 13 ... NAs werden ignoriert

auch für *mean(), sd(), apply(), aggregate(),...*

beim Daten einlesen:

> *x <- read.table('data.txt', **na.strings=...**)* ..angeben wie NAs im einzulesenden Datensatzkodiert sind z.B. „-999', „NA'...

oder nach einlesen mit *replace(x, x==...,NA)*

Umgang mit Fehlwerten in R

UNIVERSITÄT BERN
OESCHGER CENTRE CLIMATE CHANGE RESEARCH

ABER: Abfrage von Fehlwerten mit „is.na'

```
> a <- c(1, 3, 4, NA, 5)
> is.na(a)
      [1] FALSE FALSE FALSE  TRUE FALSE
> which(is.na(a))
      [1] 4
```

...**nicht** a==NA

...logischer Vector

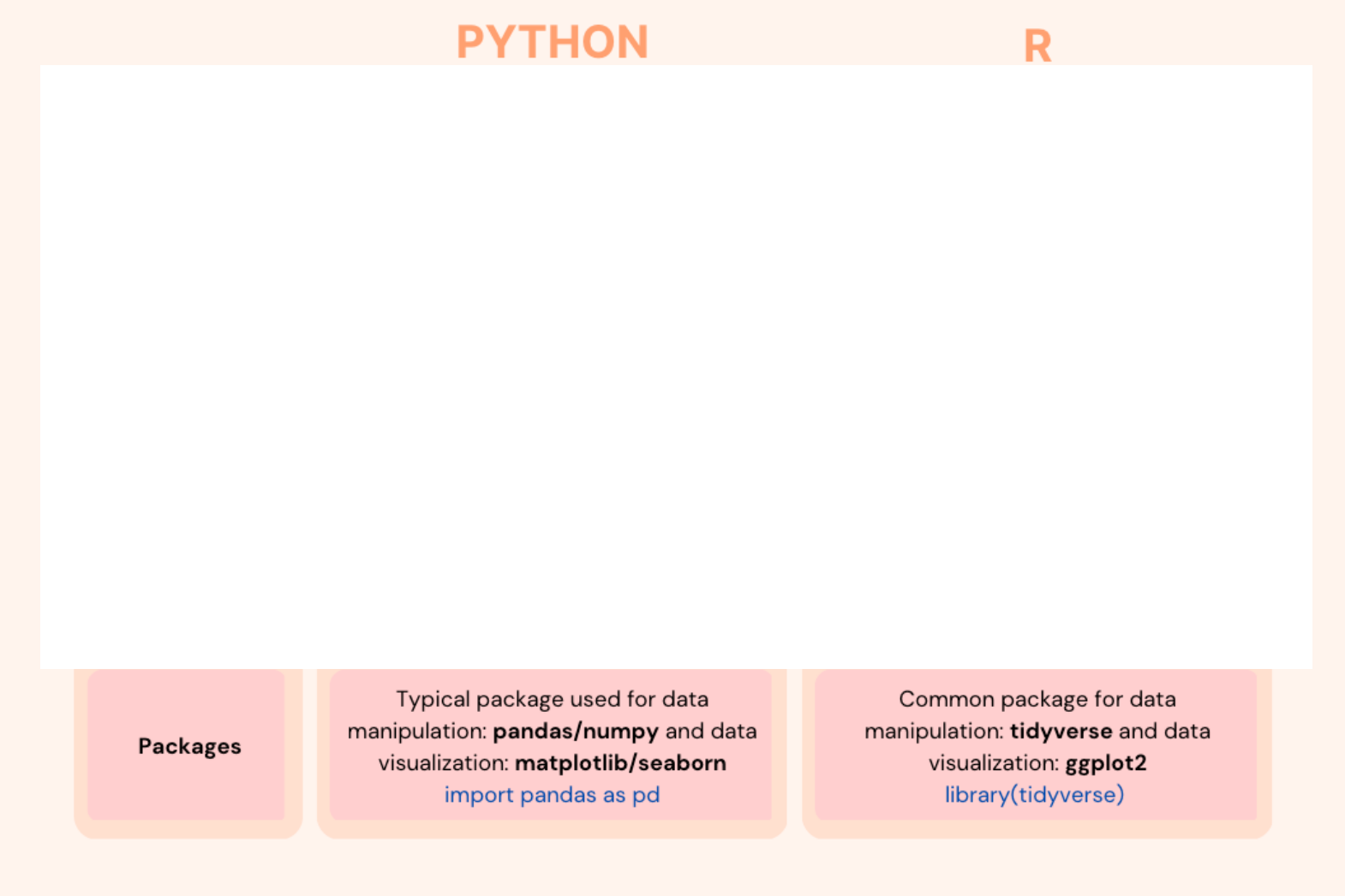
...Vektor mit Elementnummer

umgekehrt

```
> !is.na(a)
[1]  TRUE  TRUE  TRUE FALSE  TRUE
> which(! is.na(a))
```

...! bedeutet NICHT

Python vs R



https://towardsdatascience.com/the-starter-guide-for-transitioning-your-python-projects-to-r-8de4122b04ad

Packages/Libraries (Pakete/Bibliotheken) installieren und laden

UNIVERSITÄT BERN
OESCHGER CENTRE CLIMATE CHANGE RESEARCH

Am Beginn ins Script schreiben welche Pakete benötigt und geladen werden sollen (z.B. library('lattice'))

Dazu ist einmalige Installation auf Computer notwendig:

> *install.packages('lattice', dependencies=TRUE)* ... nur 1 x, dann auf Computer installiert

Dann am Anfang von jedem Skript laden, welches die Funktionen der Bibliothek nutzen soll:

> *library(lattice)* ... Paket laden (in jeder Session!)

Hilfe zur Bibliothek:

> *library(help=lattice)* ... Hilfe/Information zu package

> *?lattice* ... Hilfe/Information zu package

> *demo(lattice)* ... einige packages haben demo files

Tidyverse Library

UNIVERSITÄT BERN
OESCHGER CENTRE CLIMATE CHANGE RESEARCH

```
> install.packages("tidyverse")
> library("tidyverse")
```

Objekt type “tibble” einlesen, ähnlich data.frame()

```
> saison=read_csv("meteodaten_saison.csv")
```

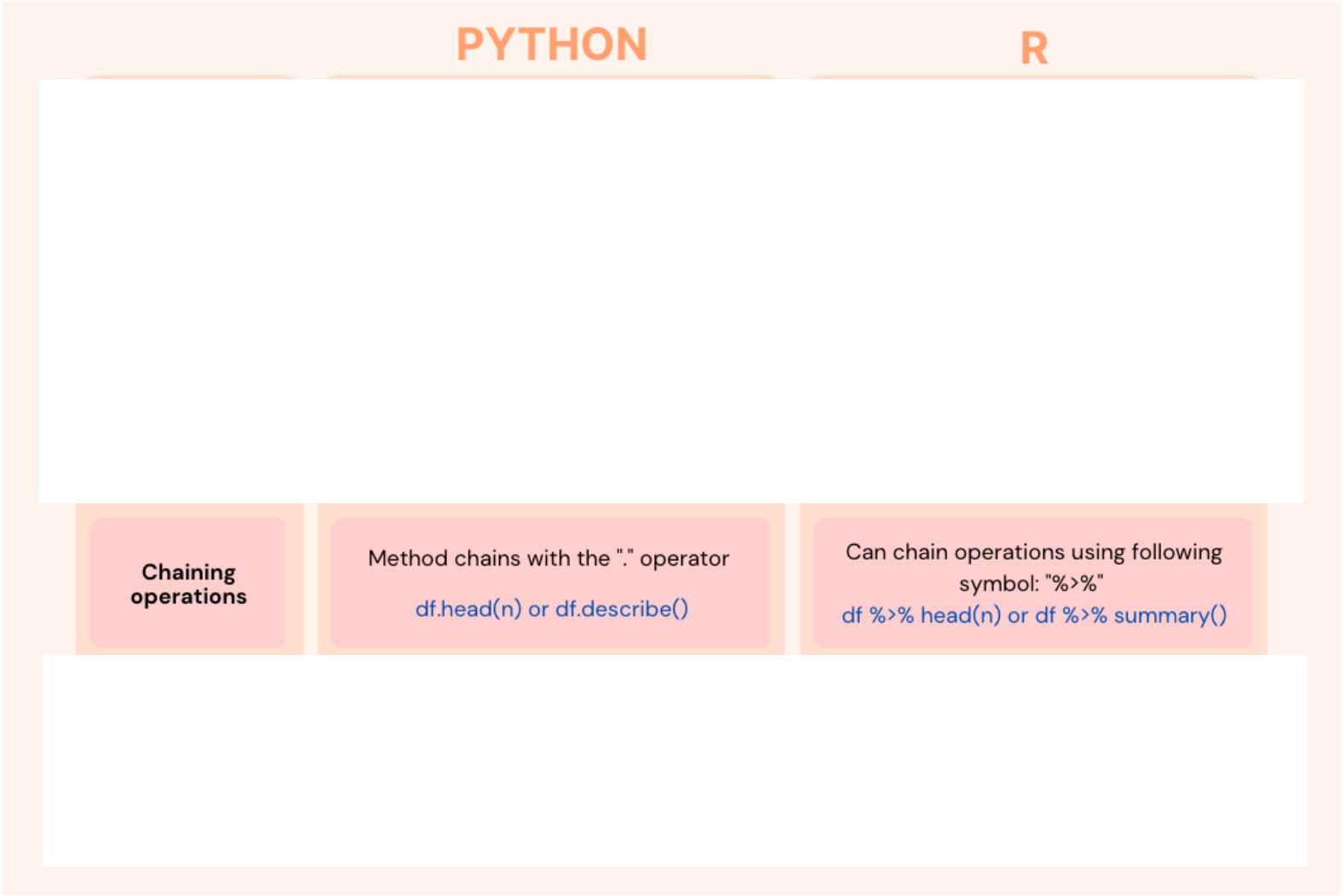
Datentransformationen

```
> winter <- filter(saison, Saison == "Winter(DJF)") # auswählen
> arrange(saison, desc(Jahr))          # sortieren
> select(saison, Jahr, Saison, Genf_Mitteltemperatur)
```

Pipes

```
> saison = saison %>% mutate(temp_diff_genf_san_bern = Genf_Mitteltemperatur -
  GrStBernhard_Mitteltemperatur)
```

Python vs R



https://towardsdatascience.com/the-starter-guide-for-transitioning-your-python-projects-to-r-8de4122b04ad

ggplot()

UNIVERSITÄT BERN

OESCHGER CENTRE
CLIMATE CHANGE RESEARCH

```
> ggplot(data = winter) +
        geom_point(mapping = aes(x = Jahr, y = Genf_Mitteltemperatur))
        # Fügt Punkte hinzu
```

Weitere “aes” Parameter:

color = Saison ; size = ... ; alpha = ... ; shape = ...

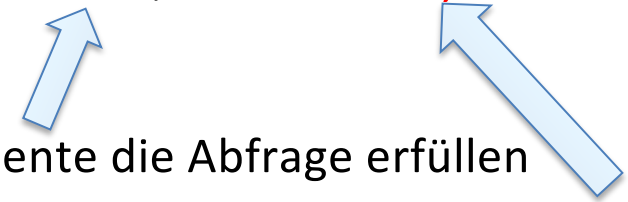
Mehrere Abbildungen kombinieren:

```
> ggplot(data = saison) +
>  geom_point(mapping = aes(x = Jahr, y = Genf_Mitteltemperatur, color=Saison)) +
>  facet_wrap(~ Saison, nrow = 2, ncol = 2)
```

ifelse – Bedingte Anweisung für Vektoren

UNIVERSITÄT BERN
OESCHGER CENTRE
CLIMATE CHANGE RESEARCH

> ifelse (Abfrage , wenn TRUE, wenn FALSE *)*



Operation für alle Elemente die Abfrage erfüllen

Operation für alle Elemente die Abfrage NICHT erfüllen

z.B.:

```
> a <- c( 5, 7, 10)
> ifelse ( a==7 ,  a+1, a-1 )
> [1]  4  8  9
```

if – Bedingte Anweisung
für einzelne Elemente, keine Vektoren

UNIVERSITÄT BERN
OESCHGER CENTRE
CLIMATE CHANGE RESEARCH

> if (Bedingung *) {* was ausführen wenn Bedingung TRUE *}*

> if (Bedingung *) {*
was ausführen wenn Bedingung TRUE
} else {
was ausführen wenn Bedingung FALSE
}

z.B.:

```
> b <- 5
> if ( b==10 ) { b <- b+1 } else { b <- b-1 }
> b
[1] 4
```

Schleifen (loops): z.B. *for() – Schleife*

UNIVERSITÄT BERN
OESCHGER CENTRE CLIMATE CHANGE RESEARCH

wenn gleiche Operationen mehrmal vorgenommen werden

müssen z. B. in Algorithmen

*> for (*beliebiger Name *in* Vektor angeben*) { Operationen }*

für alle „beliebiger Name' (z.B. „i') aus dem angegebenen „Vektor' wird nacheinander eine Operation durchgeführt

z.B.:

> for (i in 3:5) { print (i+1) }

[1] 4

[1] 5

[1] 6

Schleifen (loops): z.B. *for() – Schleife mit Laufindex*

UNIVERSITÄT BERN
OESCHGER CENTRE CLIMATE CHANGE RESEARCH

z.B.:

```
> x <- c (8, 10, 12)
> xx <- vector()          # leeren Vektor für Ergebnisse definieren

> j=1                     # laufender Index (ausserhalb Schleife!)
> for ( i in x) {
    xx [j] <- ( i+1 )
    j=j+1
  }

> xx
[1] 9  11  13
> j
[1] 4
```

eigene Funktionen I

UNIVERSITÄT BERN

OESCHGER CENTRE
CLIMATE CHANGE RESEARCH

> Funktionsname <- function (Argument 1, Argument 2, ...) Ausdruck

> Funktionsname <- function (Argument 1, Argument 2, ...){

mehrere Ausdrücke

...

}

eigene Funktionen II

UNIVERSITÄT BERN
OESCHGER CENTRE CLIMATE CHANGE RESEARCH

z.B.: Funktion die 2 Werte multipliziert

```
> myFunc <- function (x, y) {
        w <- x * y
        return(w)        # return() gibt Ergebnis aus
  }

> myFunc(1,2)            # Funktion aufrufen mit Argumenten
[1]  2
```

eigene Funktionen III

UNIVERSITÄT BERN
OESCHGER CENTRE CLIMATE CHANGE RESEARCH

z.B.: Funktion die nur alle positiven Werte eines Vektors ausgibt

```
> myFunc2 <- function (x) {
        y <- x[x>0]
        return(y)          # return() gibt Ergebnis aus
  }

> myFunc2(c(1,2,-1,4))     # Funktion aufrufen mit Argumenten
[1]  1  2  4
```

!! Funktionen kann man in eigenem R-script speichern ('auslagern') und mit source() in Arbeitsskript laden

Python vs R

UNIVERSITÄT BERN
OESCHGER CENTRE
CLIMATE CHANGE RESEARCH

	PYTHON	R
Assigning a Variable	Use assignment operator "=" x = 1	Use assignment operator "<-" or "=" x <- 1 or x = 1
Data Structure Difference	Python list datatype is equivalent to vector datatype in R ex_list = [1, 2, 3, 4, 5]	Common data type in R is a vector ex_vector <- c(1, 2, 3, 4, 5)
Indexing and slicing	Indexing starts from 0, inclusive of the start index and excludes the end index. ex_list[0] and ex_list[0:2] *output: 1* *output: [1, 2]*	Indexing starts from 1, Indexing is inclusive of both start and end index ex_vector[1] and ex_vector[1:2] *output: 1* *output: 1, 2*
Chaining operations	Method chains with the "." operator df.head(n) or df.describe()	Can chain operations using following symbol: "%>%" df %>% head(n) or df %>% summary()
Packages	Typical package used for data manipulation: **pandas/numpy** and data visualization: **matplotlib/seaborn** import pandas as pd	Common package for data manipulation: **tidyverse** and data visualization: **ggplot2** library(tidyverse)

https://towardsdatascience.com/the-starter-guide-for-transitioning-your-python-projects-to-r-8de4122b04ad

Chatbots

UNIVERSITÄT
BERN

OESCHGER CENTRE
CLIMATE CHANGE RESEARCH

Chatbot Einsatz an Uni allgemein: Was habt ihr schon lernt und selber gemacht?

Chatbots

UNIVERSITÄT BERN

OESCHGER CENTRE
CLIMATE CHANGE RESEARCH

- Besser Fragen stellen lassen nachdem ein virtueller Gesprächspartner definiert wurde als Antworten geben lassen
- Brainstorming
- Text überarbeiten lassen, auch Sprachnivieau, Stil, etc.
- immer kontrollieren, ihr seid verantwortlich!

Chatbots - Prompting

UNIVERSITÄT BERN

OESCHGER CENTRE
CLIMATE CHANGE RESEARCH

> • Kontext: Konkretisieren Sie, in welchem Zusammenhang Ihre Anfrage steht. Durch zusätzliche Informationen können Genauigkeit und Qualität der vom Chatbot generierten Ausgaben verbessert werden.

> • Zielgruppe: Identifizieren Sie die Zielgruppe und berücksichtigen Sie deren Kenntnisse und Erfahrungen. Geben Sie diese Informationen in klarer und verständlicher Sprache an.

> • Zielsetzung: Beschreiben Sie präzise, welches Ziel mit dem Prompt erreicht werden soll und überlegen Sie sich, welche Art von Informationen der Chatbot hierfür benötigt.

> • Datenquelle: Sollten Sie zusätzliche Daten eingeben, stellen Sie sicher, dass die Daten von hoher Qualität und Präzision sind.

https://lead.unibe.ch/dienstleistungen/chatbots_in_der_hochschullehre/prompting___der_zielfuehrende_dialog/index_ger.html

Chatbots - Programmieren

UNIVERSITÄT BERN
OESCHGER CENTRE
CLIMATE CHANGE RESEARCH

1. Interaktion mit einem Chatbot

> Fragen zum Programmieren, Funktionen oder Syntax stellen

> Beispiel: “Wie erstelle ich eine Weltkarte mit Kontinentgrenzen in R?” oder “Was entspricht einem Python dictionary in R?”

2. Chatbots um Fehlermeldungen zu verstehen

> Fehlermeldungen oder Code der Problem verursacht einfügen und nach Hilfe fragen.

> Dies führt meist zu leichter verständlichen Fehlerbeschreibungen.

3. Lernen

> Chatbot nach Empfehlungen für gutes Codieren fragen oder nach effizienterer Variante.

UNIVERSITÄT
BERN

OESCHGER CENTRE
CLIMATE CHANGE RESEARCH

ÜBUNGEN 3

Übungen 3.1

UNIVERSITÄT BERN
OESCHGER CENTRE
CLIMATE CHANGE RESEARCH

- Bereche die Sommer (JJA) Temperaturanomalien zur Referenzperiode 1961 bis 1990 in Bern mit Excel.
- Schreibe R Code, um die gleiche Berechnung durchzuführen.
- Lass Dir mit einem Chatbot den R Code schreiben,
- Neuerdings können Chatbots direkt Datenanalysen ohne Programmierkenntnisse durchführen:
  https://help.openai.com/en/articles/8437071-data-analysis-with-chatgpt
  Überprüfe die Ergebnisse und den generierten Code.
- Diskutiere die Vorteile, Nachteile und Risiken aller vier Methoden.

Übungen 3

mit Standard R (Beispiele unten) oder
tidyverse und ggplot mit Chatbot Hilfe

UNIVERSITÄT BERN

OESCHGER CENTRE
CLIMATE CHANGE RESEARCH

3.2) Klimadiagramm

> Ladet den Datensatz meteodaten_tag.csv nach dem Excel Export in R **(ACHTUNG:** NA-Werte sind sowohl mit '-' als auch mit 'NA') kodiert, deshalb: na.strings = c('-','NA'))

> Mit str() ansehen, ob Daten korrekt (z.B. als **num**erisch) gelesen wurden.

> Erstellt ein Histogramm (hist())mit den Tagestemperaturen mit feinen Abständen (breaks=40).

> Wie sieht die Verteilung nach Augenmass aus?

> Berechnet die Monatsmittelwerte der Temperatur und der Bewölkung über alle Jahre (also Mittel über alle Jan, alle Feb,… wie in Klimadiagrammen). Achtung: Fehlwerte vorhanden!

> Erstellt in eine Abbildung mit zwei Barplots der Ergebnisse übereinander (par(mfrow=c(2,1))). Was erwartet ihr?

Übungen 3

UNIVERSITÄT BERN
OESCHGER CENTRE CLIMATE CHANGE RESEARCH

3.3) Boxplots

> Wählt den Zeitraum **2000-2001** in den täglichen Daten.

> z.B. *zeit <- meteodaten_tag[,1] >= 2000 & meteodaten_tag[,1]<= 2001*

> Stellt die Temperaturen dieses Zeitraumes als Funktion der Bewölkung in einem *boxplot()* dar (je einen Boxplot pro Bewölkungsklasse).

> Beschriftet die Achsen und vergebt einen Titel.

> Unter welchen Bewölkungs-bedingungen ist die Spannweite und Varianz der Temperatur am grössten?

> Findet heraus welcher Monat im Mittel der bewölkungsärmste und der -reichste Monat ist (im Mittel der 2 Jahre). Wieviel Bewölkung gibt es im Mittel in diesen Monaten (in Octas)?

Übungen 3

UNIVERSITÄT BERN
OESCHGER CENTRE
CLIMATE CHANGE RESEARCH

3.4) R als GIS Ersatz

> Installiert das Paket ' maps' und ladet es in R (z.B. library(maps)). Findet die x,y-Koordinaten von Genf und dem Gr. S. Bernhard heraus (ungefähr).

> Versucht eine Europakarte herzustellen und Genf und G.S.Bernhard als Punkte auf der Karte zu plotten und die Punkte mit Stationsnamen zu versehen:

zuerst einen leeren plot erstellen mit Koordinaten von Europa.

> *plot(x=c(-5,30), y=c(35,60),*
> *type='n', xlab='lon', ylab='lat')*

 # type=' n“ heisst es wird nichts geplottet

> *map(“world”,add=TRUE)*

 # jetzt die Stationen als Punkte dazu (evtl. in verschieden Farben, mit unterschiedlichen Symbolen und mit Text!? Verwendet Google!!)

> *points(….*

b
UNIVERSITÄT
BERN

OESCHGER CENTRE
CLIMATE CHANGE RESEARCH

Take-home messages

UNIVERSITÄT BERN
OESCHGER CENTRE
CLIMATE CHANGE RESEARCH

- Insbesondere für grosse Datensätze viel besser als Tabellenkalkulation
- Erzeugt mit einer Zeile Code grundlegende statistische Auswertungen und Abbildungen wie Histogramme und Test, die mit Excel komplizert oder gar nicht möglich sind
- Bei Fragen Internetsuchmaschine oder Chatbot gebrauchen
- Programmieren muss gar nicht so komplizert sein
- ACHTUNG: man bekommt (fast) immer etwas heraus, stellt sicher, dass es auch das Richtige ist!
- In der Prüfung müsst ihr nicht alle Funktionen und deren Syntax in R kennen, aber einfachen Code verstehen können und beschreiben können wie ein Problem lösen würdet, z.B. Scheife über alle Stationen, um statistischen Test an allen Orten durchzuführen.

Beispiel Prüfungsfrage

UNIVERSITÄT BERN
OESCHGER CENTRE CLIMATE CHANGE RESEARCH

> Was berechnet folgende R Funktion (Modus, Mittelwert, Median, Minimum oder Maximum):

```
# Funktion zur Berechnung des ??? eines Vektors
berechne_? <- function(vektor) {
# Vektor sortieren
  sortierter_vektor <- sort(vektor)
# Länge des Vektors
  n <- length(sortierter_vektor)
# ? berechnen
  if (n %% 2 == 1) { #
    ergebnis <- sortierter_vektor[(n + 1) / 2]
  } else {
    ergebnis <- (sortierter_vektor[n / 2] + sortierter_vektor[n / 2 + 1]) / 2 }
  return(ergebnis)
}
```