How I Integrate ChatGPT into My Day-to-Day Activities as a Data Scientist

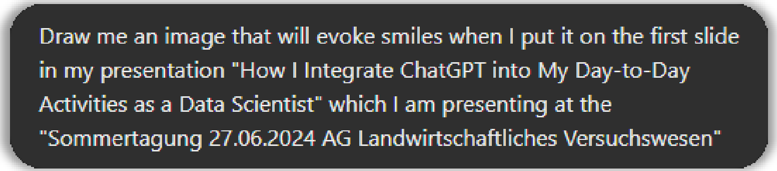
Paul Schmidt

Sommertagung 27.06.2024

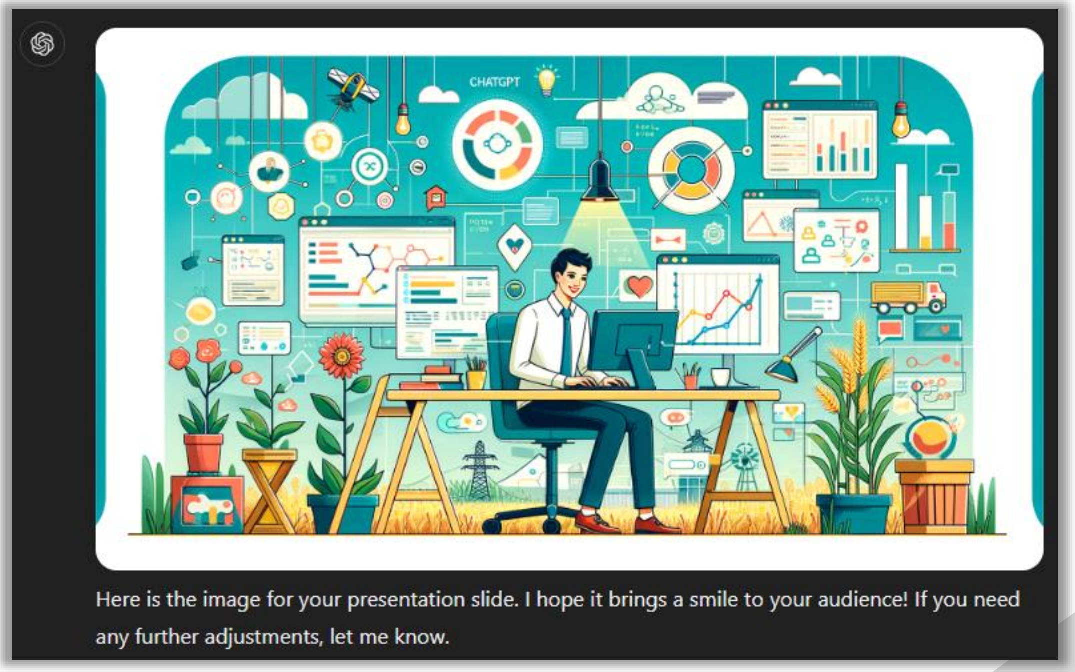
AG Landwirtschaftliches Versuchswesen

This talk is not about

- what ChatGPT is and how it works
- whether your are allowed use ChatGPT on your computer at work
- how your job becomes obsolote because AI replaces you
- how AI is useless because it once answered something incorrect and thus you cant rely on it



Draw me an image that will evoke smiles when I put it on the first slide in my presentation "How I Integrate ChatGPT into My Day-to-Day Activities as a Data Scientist" which I am presenting at the "Sommertagung 27.06.2024 AG Landwirtschaftliches Versuchswesen"



Here is the image for your presentation slide. I hope it brings a smile to your audience! If you need any further adjustments, let me know.

ChatGPT can write drafts for me

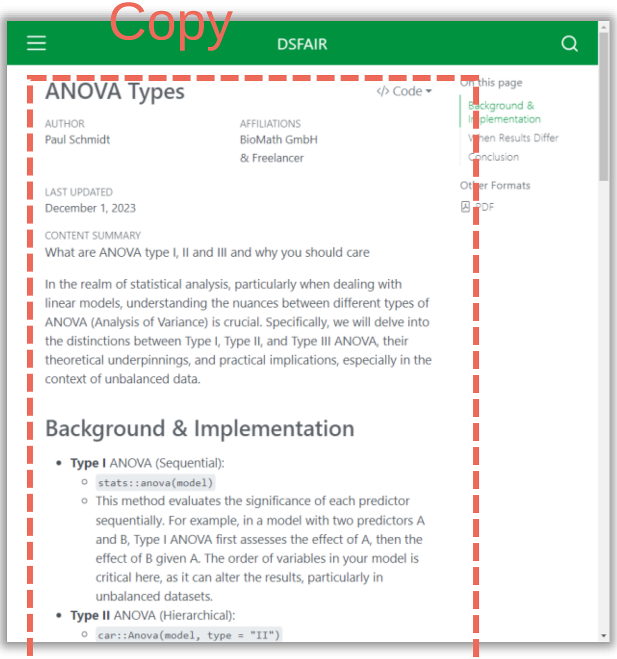
Hey Paul,

Copy

You briefly mentioned "Type III ANOVA" the other day. What do you mean by that? Are there other types, and what is the difference?

Best regards,
Maria Mustermann

Copy



ANOVA Types

AUTHOR
Paul Schmidt

AFFILIATIONS
BioMath GmbH
& Freelancer

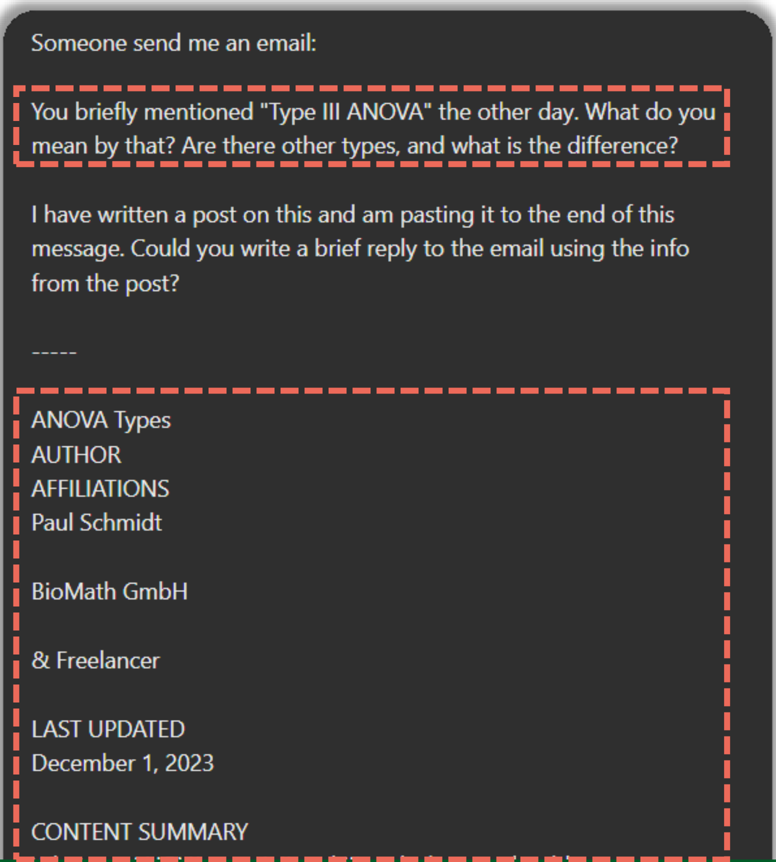
LAST UPDATED
December 1, 2023

CONTENT SUMMARY
What are ANOVA type I, II and III and why you should care

In the realm of statistical analysis, particularly when dealing with linear models, understanding the nuances between different types of ANOVA (Analysis of Variance) is crucial. Specifically, we will delve into the distinctions between Type I, Type II, and Type III ANOVA, their theoretical underpinnings, and practical implications, especially in the context of unbalanced data.

Background & Implementation

- **Type I** ANOVA (Sequential):
  - `stats::anova(model)`
  - This method evaluates the significance of each predictor sequentially. For example, in a model with two predictors A and B, Type I ANOVA first assesses the effect of A, then the effect of B given A. The order of variables in your model is critical here, as it can alter the results, particularly in unbalanced datasets.
- **Type II** ANOVA (Hierarchical):
  - `car::Anova(model, type = "II")`



Someone send me an email:

Paste

You briefly mentioned "Type III ANOVA" the other day. What do you mean by that? Are there other types, and what is the difference?

I have written a post on this and am pasting it to the end of this message. Could you write a brief reply to the email using the info from the post?

Paste

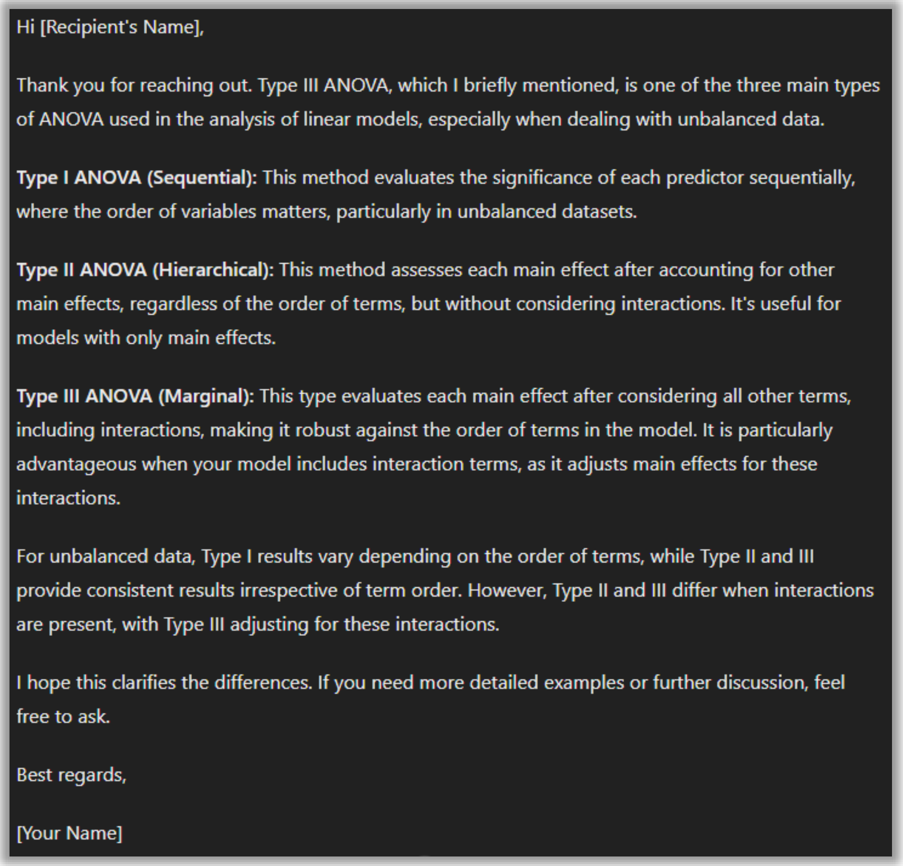
ANOVA Types
AUTHOR
AFFILIATIONS
Paul Schmidt

BioMath GmbH

& Freelancer

LAST UPDATED
December 1, 2023

CONTENT SUMMARY



Hi [Recipient's Name],

Thank you for reaching out. Type III ANOVA, which I briefly mentioned, is one of the three main types of ANOVA used in the analysis of linear models, especially when dealing with unbalanced data.

Type I ANOVA (Sequential): This method evaluates the significance of each predictor sequentially, where the order of variables matters, particularly in unbalanced datasets.

Type II ANOVA (Hierarchical): This method assesses each main effect after accounting for other main effects, regardless of the order of terms, but without considering interactions. It's useful for models with only main effects.

Type III ANOVA (Marginal): This type evaluates each main effect after considering all other terms, including interactions, making it robust against the order of terms in the model. It is particularly advantageous when your model includes interaction terms, as it adjusts main effects for these interactions.

For unbalanced data, Type I results vary depending on the order of terms, while Type II and III provide consistent results irrespective of term order. However, Type II and III differ when interactions are present, with Type III adjusting for these interactions.

I hope this clarifies the differences. If you need more detailed examples or further discussion, feel free to ask.

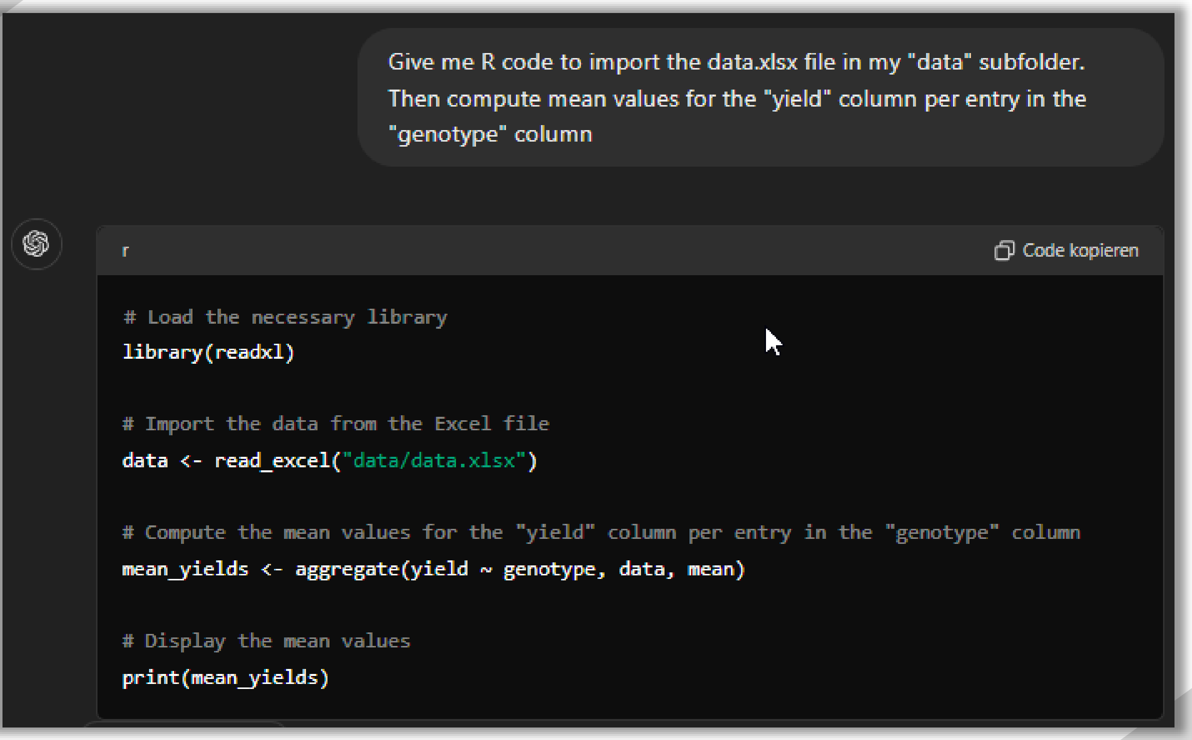
Best regards,

[Your Name]

More Examples:

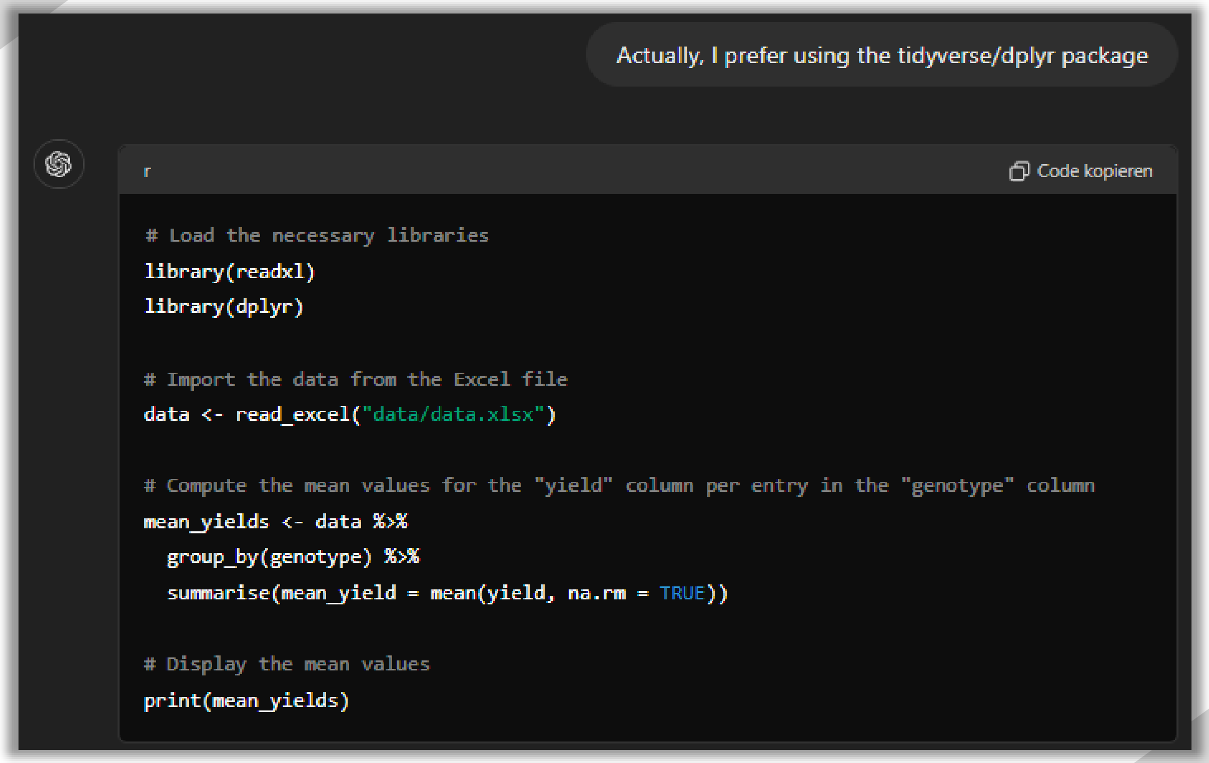
- Take my bullet points here […] and rewrite them into a proper text for my scientific report
- Give me three alternative phrasings for this paragraph […]
- Here is my last publication […], try to copy my writing style and give me an introductional paragraph for my new publication on topic […]
- Here is a call for proposals for a project […]. Summarize in bullet points what is required.

ChatGPT can „*understand*“ Code



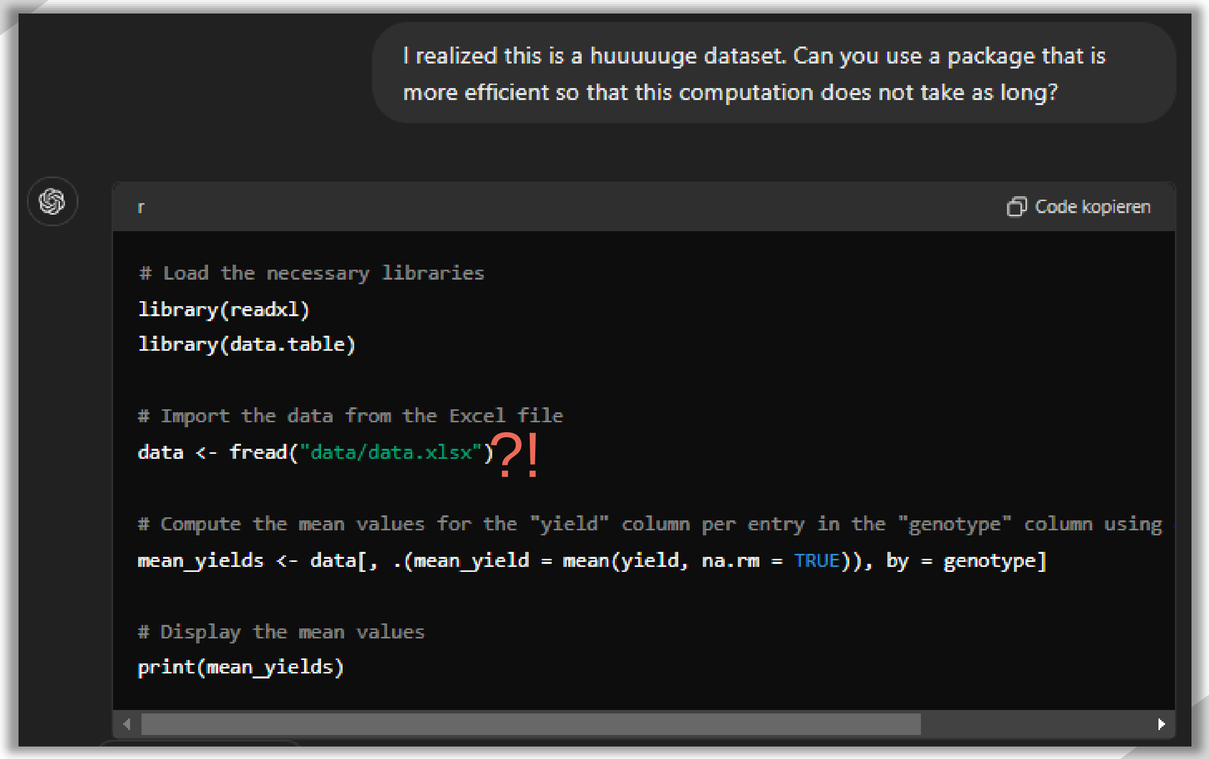
Give me R code to import the data.xlsx file in my "data" subfolder. Then compute mean values for the "yield" column per entry in the "genotype" column

```r
# Load the necessary library
library(readxl)

# Import the data from the Excel file
data <- read_excel("data/data.xlsx")

# Compute the mean values for the "yield" column per entry in the "genotype" column
mean_yields <- aggregate(yield ~ genotype, data, mean)

# Display the mean values
print(mean_yields)
```



Actually, I prefer using the tidyverse/dplyr package

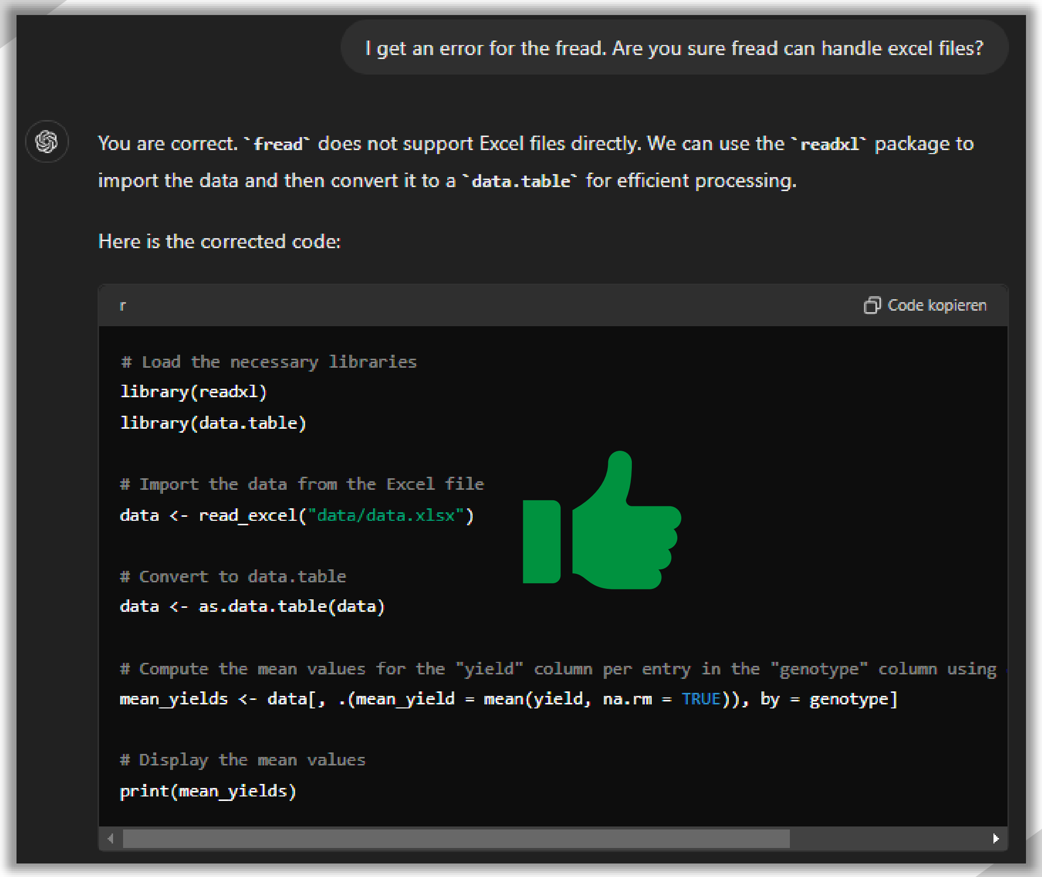
```r
# Load the necessary libraries
library(readxl)
library(dplyr)

# Import the data from the Excel file
data <- read_excel("data/data.xlsx")

# Compute the mean values for the "yield" column per entry in the "genotype" column
mean_yields <- data %>%
  group_by(genotype) %>%
  summarise(mean_yield = mean(yield, na.rm = TRUE))

# Display the mean values
print(mean_yields)
```



I realized this is a huuuuuge dataset. Can you use a package that is more efficient so that this computation does not take as long?

```r
# Load the necessary libraries
library(readxl)
library(data.table)

# Import the data from the Excel file
data <- fread("data/data.xlsx")

# Compute the mean values for the "yield" column per entry in the "genotype" column using
mean_yields <- data[, .(mean_yield = mean(yield, na.rm = TRUE)), by = genotype]

# Display the mean values
print(mean_yields)
```

?!

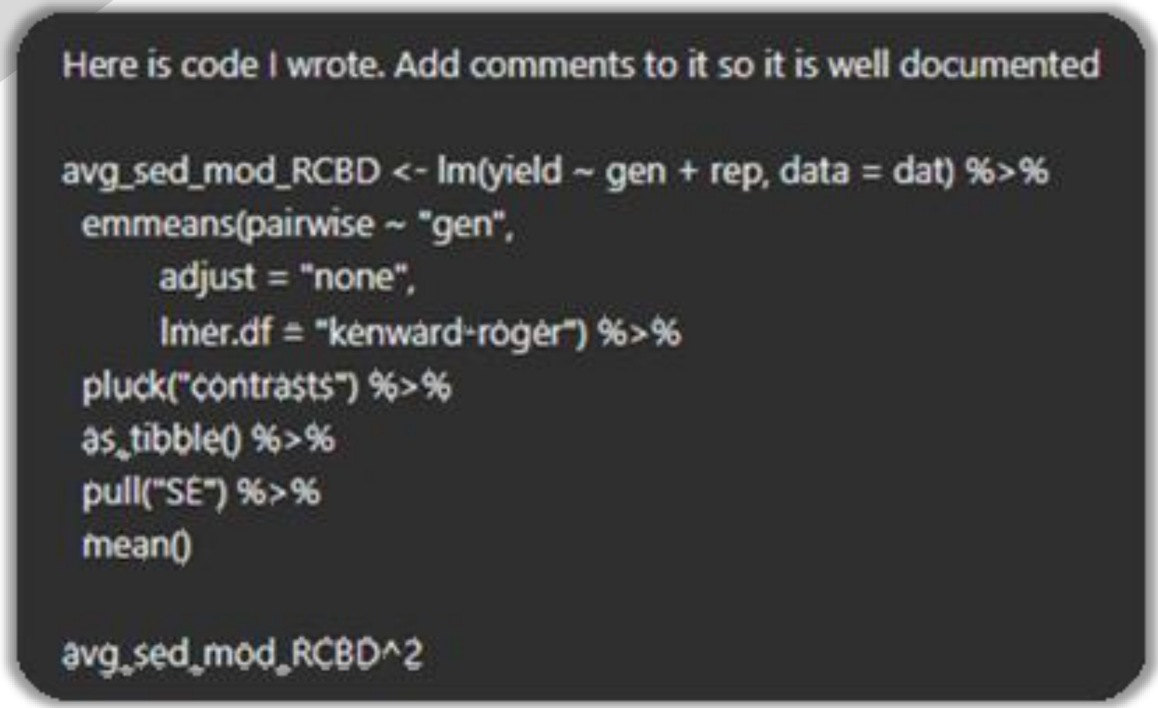


I get an error for the fread. Are you sure fread can handle excel files?

You are correct. `fread` does not support Excel files directly. We can use the `readxl` package to import the data and then convert it to a `data.table` for efficient processing.

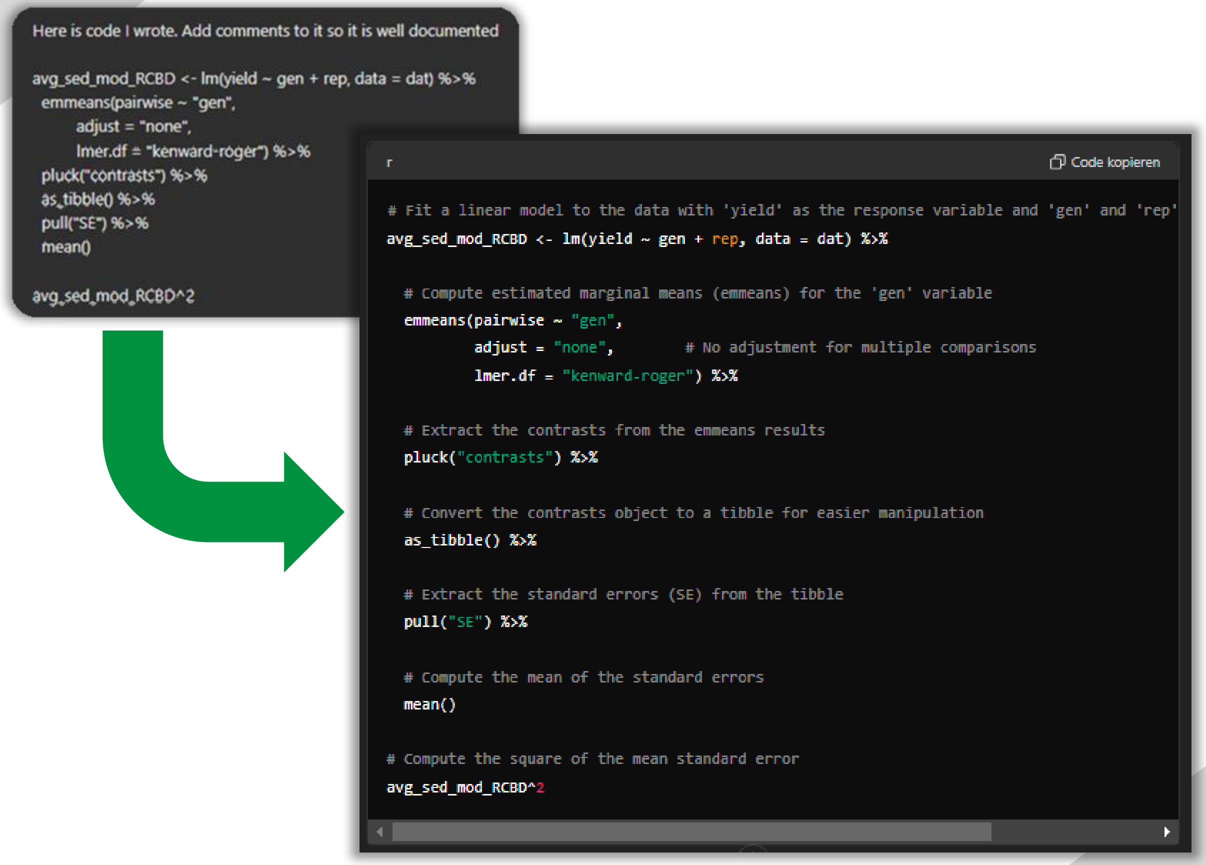
Here is the corrected code:

```r
# Load the necessary libraries
library(readxl)
library(data.table)

# Import the data from the Excel file
data <- read_excel("data/data.xlsx")

# Convert to data.table
data <- as.data.table(data)

# Compute the mean values for the "yield" column per entry in the "genotype" column using
mean_yields <- data[, .(mean_yield = mean(yield, na.rm = TRUE)), by = genotype]

# Display the mean values
print(mean_yields)
```



Here is code I wrote. Add comments to it so it is well documented

```
avg_sed_mod_RCBD <- lm(yield ~ gen + rep, data = dat) %>%
  emmeans(pairwise ~ "gen",
          adjust = "none",
          lmer.df = "kenward-roger") %>%
  pluck("contrasts") %>%
  as_tibble() %>%
  pull("SE") %>%
  mean()

avg_sed_mod_RCBD^2
```

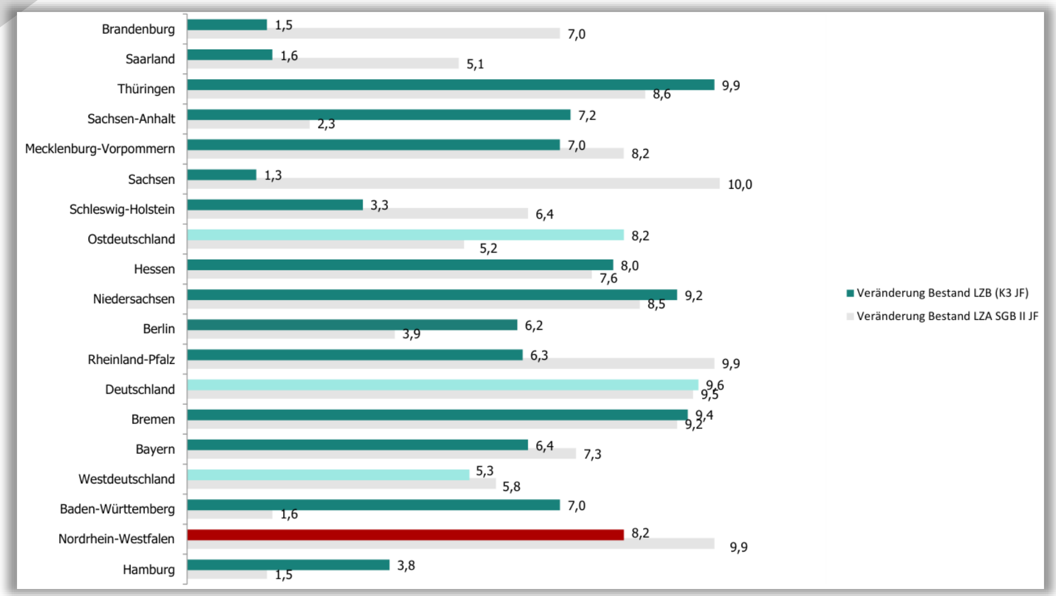


Here is code I wrote. Add comments to it so it is well documented

```
avg_sed_mod_RCBD <- lm(yield ~ gen + rep, data = dat) %>%
 emmeans(pairwise ~ "gen",
       adjust = "none",
       lmer.df = "kenward-roger") %>%
 pluck("contrasts") %>%
 as_tibble() %>%
 pull("SE") %>%
 mean()

avg_sed_mod_RCBD^2
```

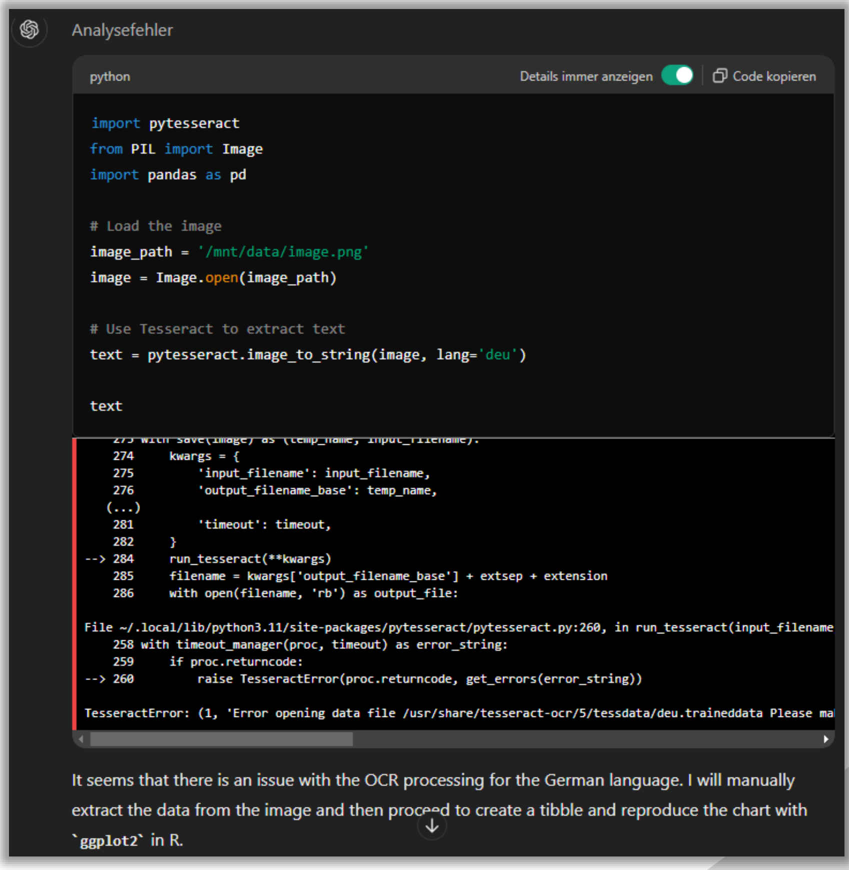
```r
# Fit a linear model to the data with 'yield' as the response variable and 'gen' and 'rep'
avg_sed_mod_RCBD <- lm(yield ~ gen + rep, data = dat) %>%

  # Compute estimated marginal means (emmeans) for the 'gen' variable
  emmeans(pairwise ~ "gen",
          adjust = "none",        # No adjustment for multiple comparisons
          lmer.df = "kenward-roger") %>%

  # Extract the contrasts from the emmeans results
  pluck("contrasts") %>%

  # Convert the contrasts object to a tibble for easier manipulation
  as_tibble() %>%

  # Extract the standard errors (SE) from the tibble
  pull("SE") %>%

  # Compute the mean of the standard errors
  mean()

# Compute the square of the mean standard error
avg_sed_mod_RCBD^2
```

ChatGPT can „*read*“ images



Extrahiere voerst die Daten aus dieser Abbildung. Dann reproduziere sie als tibble und erzeuge die Abbildung mit ggplot2

At first, it struggled a bit…

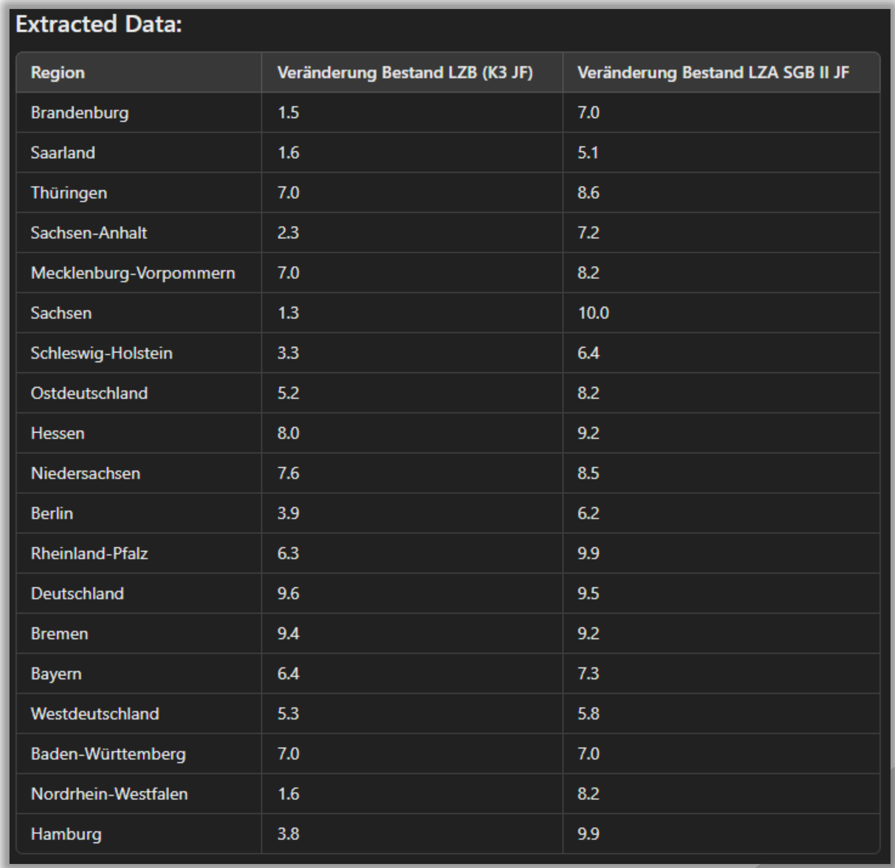


```python
import pytesseract
from PIL import Image
import pandas as pd

# Load the image
image_path = '/mnt/data/image.png'
image = Image.open(image_path)

# Use Tesseract to extract text
text = pytesseract.image_to_string(image, lang='deu')

text
```

```
    273 with save(image) as (temp_name, input_filename):
    274     kwargs = {
    275         'input_filename': input_filename,
    276         'output_filename_base': temp_name,
   (...)
    281         'timeout': timeout,
    282     }
--> 284     run_tesseract(**kwargs)
    285     filename = kwargs['output_filename_base'] + extsep + extension
    286     with open(filename, 'rb') as output_file:

File ~/.local/lib/python3.11/site-packages/pytesseract/pytesseract.py:260, in run_tesseract(input_filename
    258 with timeout_manager(proc, timeout) as error_string:
    259     if proc.returncode:
--> 260         raise TesseractError(proc.returncode, get_errors(error_string))

TesseractError: (1, 'Error opening data file /usr/share/tesseract-ocr/5/tessdata/deu.traineddata Please ma
```

It seems that there is an issue with the OCR processing for the German language. I will manually extract the data from the image and then proceed to create a tibble and reproduce the chart with `ggplot2` in R.

…but tried again

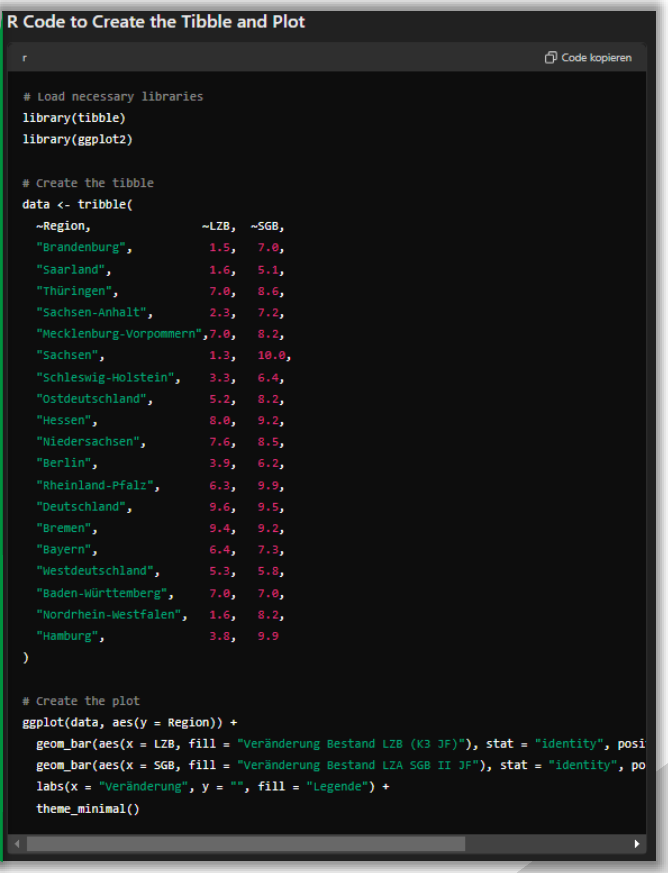


Extracted Data:

Region	Veränderung Bestand LZB (K3 JF)	Veränderung Bestand LZA SGB II JF
Brandenburg	1.5	7.0
Saarland	1.6	5.1
Thüringen	7.0	8.6
Sachsen-Anhalt	2.3	7.2
Mecklenburg-Vorpommern	7.0	8.2
Sachsen	1.3	10.0
Schleswig-Holstein	3.3	6.4
Ostdeutschland	5.2	8.2
Hessen	8.0	9.2
Niedersachsen	7.6	8.5
Berlin	3.9	6.2
Rheinland-Pfalz	6.3	9.9
Deutschland	9.6	9.5
Bremen	9.4	9.2
Bayern	6.4	7.3
Westdeutschland	5.3	5.8
Baden-Württemberg	7.0	7.0
Nordrhein-Westfalen	1.6	8.2
Hamburg	3.8	9.9

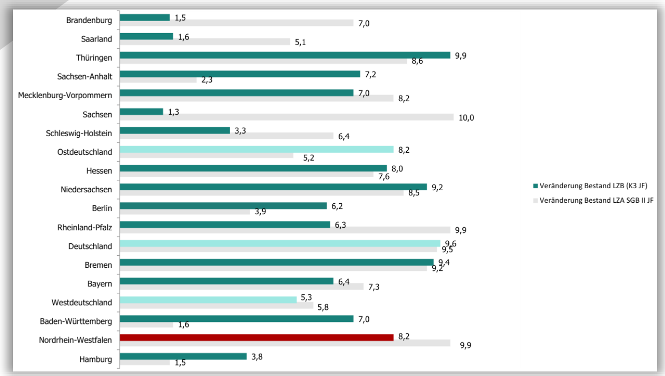
…and made it work (partially)

and so far, all this was its initial answer!



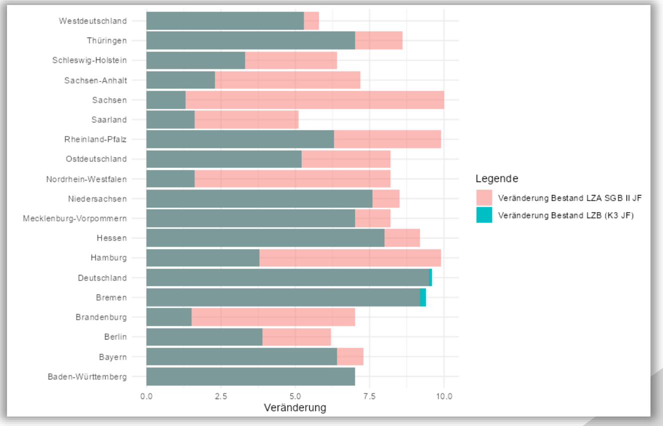
R Code to Create the Tibble and Plot

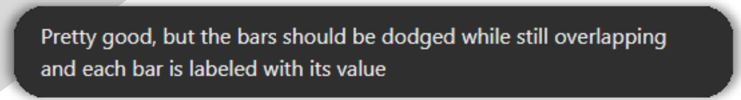
```r
# Load necessary libraries
library(tibble)
library(ggplot2)

# Create the tibble
data <- tribble(
  ~Region,                ~LZB,  ~SGB,
  "Brandenburg",           1.5,   7.0,
  "Saarland",              1.6,   5.1,
  "Thüringen",             7.0,   8.6,
  "Sachsen-Anhalt",        2.3,   7.2,
  "Mecklenburg-Vorpommern",7.0,   8.2,
  "Sachsen",               1.3,   10.0,
  "Schleswig-Holstein",    3.3,   6.4,
  "Ostdeutschland",        5.2,   8.2,
  "Hessen",                8.0,   9.2,
  "Niedersachsen",         7.6,   8.5,
  "Berlin",                3.9,   6.2,
  "Rheinland-Pfalz",       6.3,   9.9,
  "Deutschland",           9.6,   9.5,
  "Bremen",                9.4,   9.2,
  "Bayern",                6.4,   7.3,
  "Westdeutschland",       5.3,   5.8,
  "Baden-Württemberg",     7.0,   7.0,
  "Nordrhein-Westfalen",   1.6,   8.2,
  "Hamburg",               3.8,   9.9
)

# Create the plot
ggplot(data, aes(y = Region)) +
  geom_bar(aes(x = LZB, fill = "Veränderung Bestand LZB (K3 JF)"), stat = "identity", posi
  geom_bar(aes(x = SGB, fill = "Veränderung Bestand LZA SGB II JF"), stat = "identity", po
  labs(x = "Veränderung", y = "", fill = "Legende") +
  theme_minimal()
```



← Goal

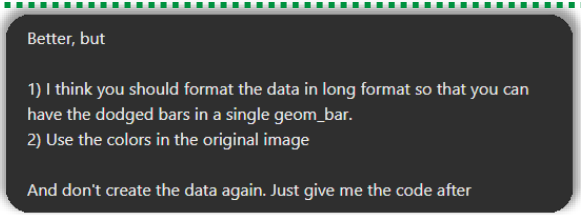
ChatGPT‘s first try →





Pretty good, but the bars should be dodged while still overlapping and each bar is labeled with its value

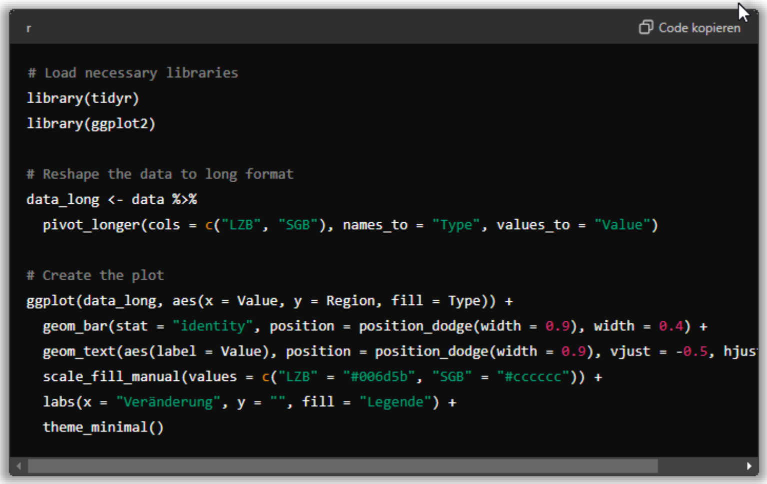
2nd try not shown



Better, but

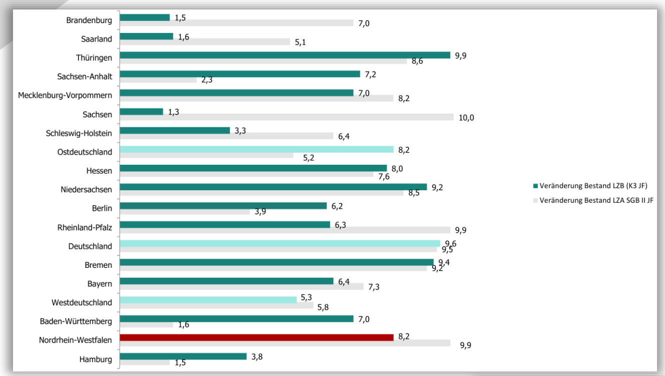
1) I think you should format the data in long format so that you can have the dodged bars in a single geom_bar.
2) Use the colors in the original image

And don't create the data again. Just give me the code after



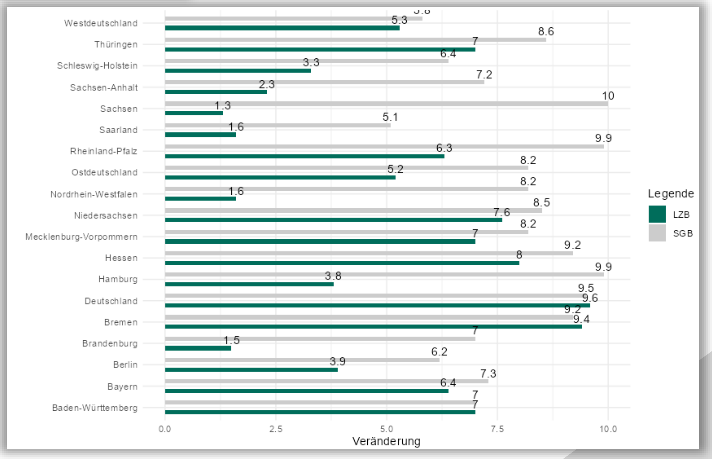
```r
# Load necessary libraries
library(tidyr)
library(ggplot2)

# Reshape the data to long format
data_long <- data %>%
  pivot_longer(cols = c("LZB", "SGB"), names_to = "Type", values_to = "Value")

# Create the plot
ggplot(data_long, aes(x = Value, y = Region, fill = Type)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.9), width = 0.4) +
  geom_text(aes(label = Value), position = position_dodge(width = 0.9), vjust = -0.5, hjus
  scale_fill_manual(values = c("LZB" = "#006d5b", "SGB" = "#cccccc")) +
  labs(x = "Veränderung", y = "", fill = "Legende") +
  theme_minimal()
```

3rd try



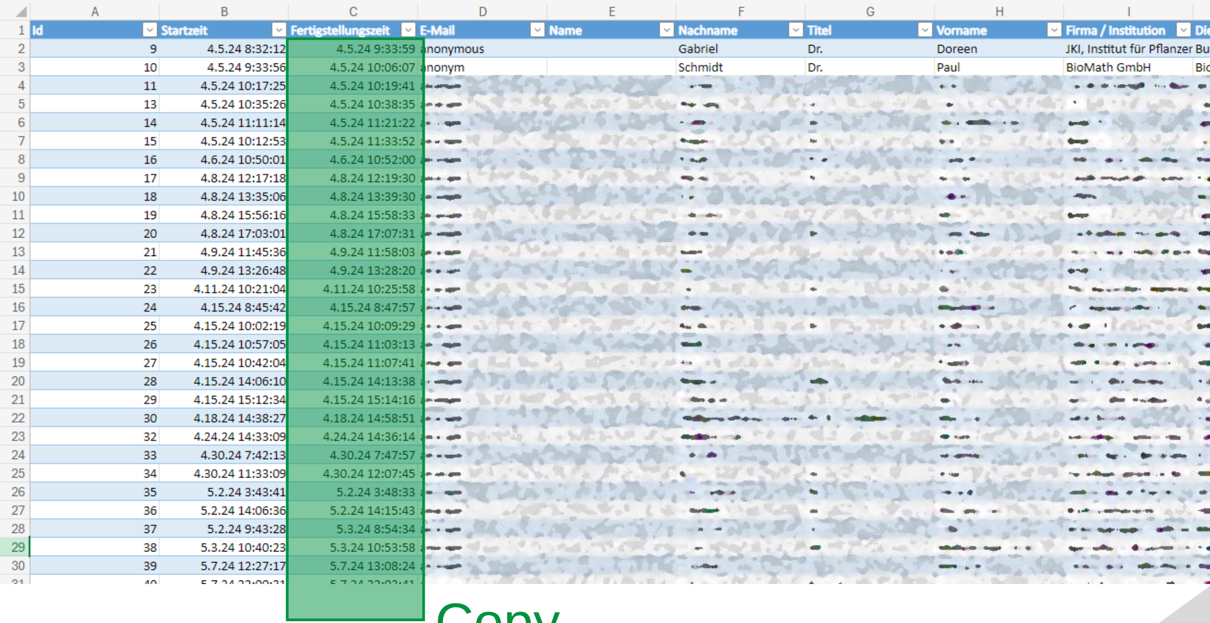
← Goal

ChatGPT‘s 3rd try →



ChatGPT can **RUN** Python Code

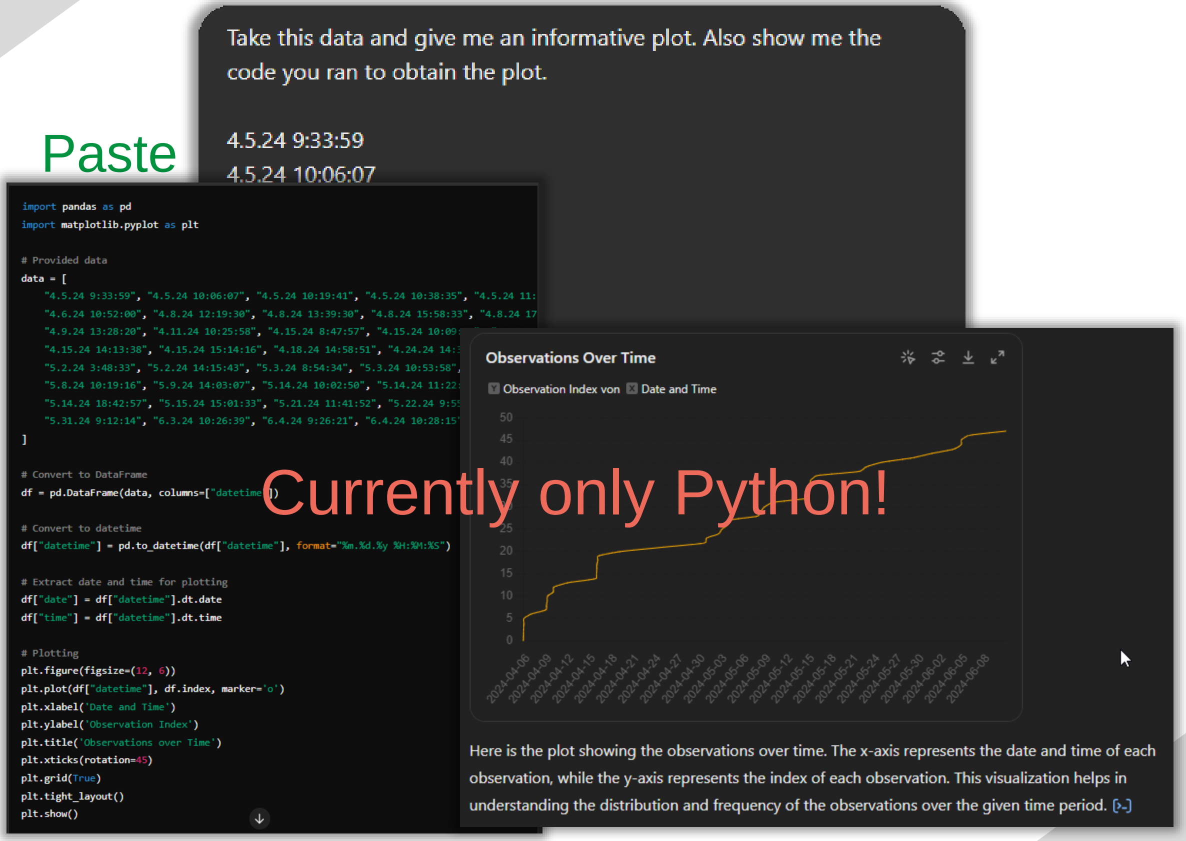
Table resulting from online registration to this event:



	A	B	C	D	E	F	G	H	I	
1	Id	Startzeit	Fertigstellungszeit	E-Mail	Name	Nachname	Titel	Vorname	Firma / Institution	Die
2	9	4.5.24 8:32:12	4.5.24 9:33:59	anonymous		Gabriel	Dr.	Doreen	JKI, Institut für Pflanzer	Bu
3	10	4.5.24 9:33:56	4.5.24 10:06:07	anonym		Schmidt	Dr.	Paul	BioMath GmbH	Bio
4	11	4.5.24 10:17:25	4.5.24 10:19:41	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
5	13	4.5.24 10:35:26	4.5.24 10:38:35	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
6	14	4.5.24 11:11:14	4.5.24 11:21:22	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
7	15	4.5.24 10:12:53	4.5.24 11:33:52	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
8	16	4.6.24 10:50:01	4.6.24 10:52:00	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
9	17	4.8.24 12:17:18	4.8.24 12:19:30	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
10	18	4.8.24 13:35:06	4.8.24 13:39:30	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
11	19	4.8.24 15:56:16	4.8.24 15:58:33	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
12	20	4.8.24 17:03:01	4.8.24 17:07:31	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
13	21	4.9.24 11:45:36	4.9.24 11:58:03	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
14	22	4.9.24 13:26:48	4.9.24 13:28:20	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
15	23	4.11.24 10:21:04	4.11.24 10:25:58	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
16	24	4.15.24 8:45:42	4.15.24 8:47:57	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
17	25	4.15.24 10:02:19	4.15.24 10:09:29	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
18	26	4.15.24 10:57:05	4.15.24 11:03:13	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
19	27	4.15.24 10:42:04	4.15.24 11:07:41	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
20	28	4.15.24 14:06:10	4.15.24 14:13:38	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
21	29	4.15.24 15:12:34	4.15.24 15:14:16	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
22	30	4.18.24 14:38:27	4.18.24 14:58:51	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
23	32	4.24.24 14:33:09	4.24.24 14:36:14	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
24	33	4.30.24 7:42:13	4.30.24 7:47:57	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
25	34	4.30.24 11:33:09	4.30.24 12:07:45	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
26	35	5.2.24 3:43:41	5.2.24 3:48:33	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
27	36	5.2.24 14:06:36	5.2.24 14:15:43	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
28	37	5.2.24 9:43:28	5.3.24 8:54:34	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
29	38	5.3.24 10:40:23	5.3.24 10:53:58	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
30	39	5.7.24 12:27:17	5.7.24 13:08:24	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]

Copy

Paste



Take this data and give me an informative plot. Also show me the code you ran to obtain the plot.

4.5.24 9:33:59
4.5.24 10:06:07

```python
import pandas as pd
import matplotlib.pyplot as plt

# Provided data
data = [
    "4.5.24 9:33:59", "4.5.24 10:06:07", "4.5.24 10:19:41", "4.5.24 10:38:35", "4.5.24 11:
    "4.6.24 10:52:00", "4.8.24 12:19:30", "4.8.24 13:39:30", "4.8.24 15:58:33", "4.8.24 17
    "4.9.24 13:28:20", "4.11.24 10:25:58", "4.15.24 8:47:57", "4.15.24 10:09:
    "4.15.24 14:13:38", "4.15.24 15:14:16", "4.18.24 14:58:51", "4.24.24 14:
    "5.2.24 3:48:33", "5.2.24 14:15:43", "5.3.24 8:54:34", "5.3.24 10:53:58",
    "5.8.24 10:19:16", "5.9.24 14:03:07", "5.14.24 10:02:50", "5.14.24 11:22:
    "5.14.24 18:42:57", "5.15.24 15:01:33", "5.21.24 11:41:52", "5.22.24 9:5
    "5.31.24 9:12:14", "6.3.24 10:26:39", "6.4.24 9:26:21", "6.4.24 10:28:15"
]

# Convert to DataFrame
df = pd.DataFrame(data, columns=["datetime"])

# Convert to datetime
df["datetime"] = pd.to_datetime(df["datetime"], format="%m.%d.%y %H:%M:%S")

# Extract date and time for plotting
df["date"] = df["datetime"].dt.date
df["time"] = df["datetime"].dt.time

# Plotting
plt.figure(figsize=(12, 6))
plt.plot(df["datetime"], df.index, marker='o')
plt.xlabel('Date and Time')
plt.ylabel('Observation Index')
plt.title('Observations over Time')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```

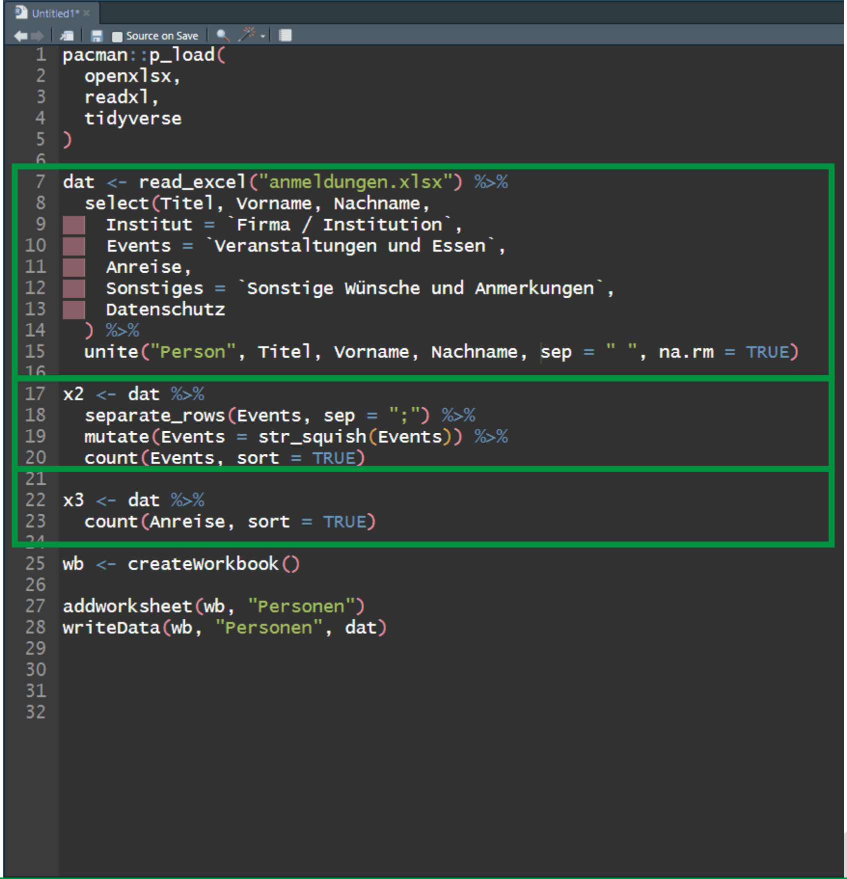
Here is the plot showing the observations over time. The x-axis represents the date and time of each observation, while the y-axis represents the index of each observation. This visualization helps in understanding the distribution and frequency of the observations over the given time period. [>_]

Currently only Python!

GitHub Copilot is like ChatGPT living inside RStudio

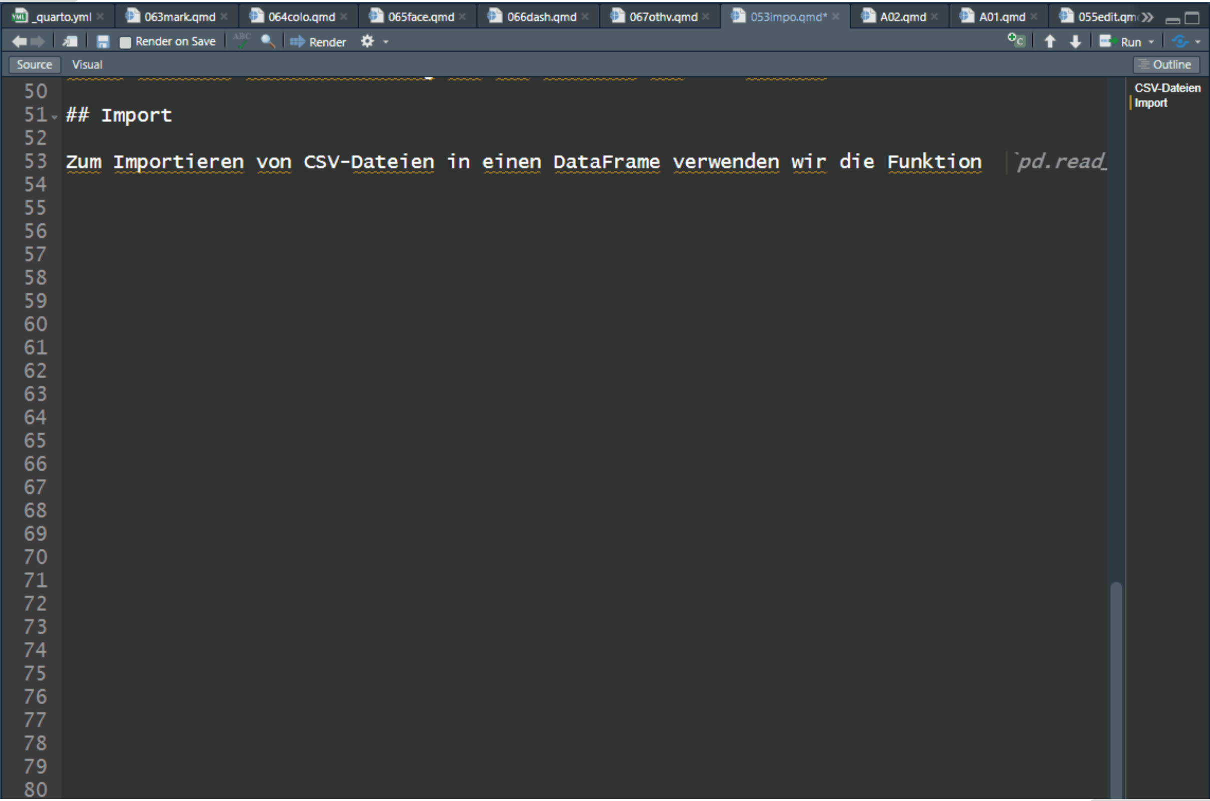
And it constantly reads all files in your project folder as well as objects in your environment. It then uses this information to always suggest what you should write next.

Example: R-Script



```r
pacman::p_load(
  openxlsx,
  readxl,
  tidyverse
)

dat <- read_excel("anmeldungen.xlsx") %>%
  select(Titel, Vorname, Nachname,
    Institut = `Firma / Institution`,
    Events = `Veranstaltungen und Essen`,
    Anreise,
    Sonstiges = `Sonstige Wünsche und Anmerkungen`,
    Datenschutz
  ) %>%
  unite("Person", Titel, Vorname, Nachname, sep = " ", na.rm = TRUE)

x2 <- dat %>%
  separate_rows(Events, sep = ";") %>%
  mutate(Events = str_squish(Events)) %>%
  count(Events, sort = TRUE)

x3 <- dat %>%
  count(Anreise, sort = TRUE)

wb <- createWorkbook()

addworksheet(wb, "Personen")
writeData(wb, "Personen", dat)
```

Example: Quarto Document (Rmarkdown successor)



```
## Import

Zum Importieren von CSV-Dateien in einen DataFrame verwenden wir die Funktion `pd.read_
```

BioMath

APPLIED STATISTICS AND INFORMATICS
IN LIFE SCIENCES

www.biomath.de

central@biomath.de

☎ +49 172 3091577

Ask me about how ChatGPT helped me improve this map I created with {osmdata} & {ggplot2} in R

BioMath GmbH
Eppendorfer Weg 255
20251 Hamburg
Außenalster