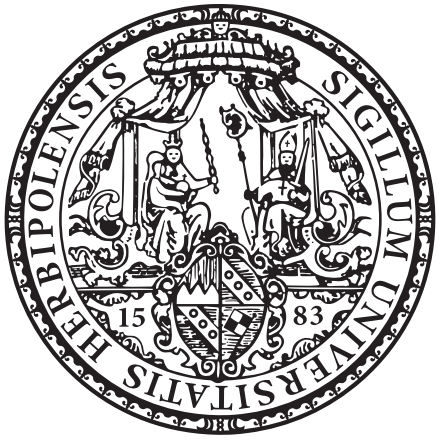


Development and Semi-Automated Analysis of an *in vitro* Dissemination Model for Myeloma Cells Interacting with Mesenchymal Stromal Cells

Entwicklung und semi-automatisierte Analyse eines *in vitro* Modells für die Disseminierung von Myelomzellen in Interaktion mit mesenchymalen Stromazellen

---

Doctoral Thesis for a Doctoral Degree

at the

GRADUATE SCHOOL OF LIFE SCIENCES,
JULIUS-MAXIMILIANS-UNIVERSITÄT WÜRZBURG,
SECTION BIOMEDICINE

submitted by

Martin Kuric

from

Bad Neustadt a.d. Saale

Würzburg, 2024

Submitted on: .....
Office stamp

Members of the *Promotionskomitee*:

Chairperson:	Prof. Dr. Uwe Gbureck
Primary Supervisor:	Prof. Dr. rer. nat. Regina Ebert
Supervisor (Second):	Prof. Dr. med. Franziska Jundt
Supervisor (Third):	Prof. Dr. rer. nat. Torsten Blunk

Date of Public Defense: .....

Date of Receipt of Certificates: .....

This work was conducted at the Department of Musculoskeletal Tissue Regeneration (Bernhard-Heine-Centre for Locomotive Research), University of Würzburg from 08.04.2018 to 31.03.2024 under the supervision of Prof Dr. rer. nat. Regina Ebert.

Acknowledgements

Lorem Ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum

Summary

Lorem Ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum

Zusammenfassung

Lorem Ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum bla

Contents

Introduction

To provide a comprehensive background for the following chapters that focus on the interaction of human mesenchymal stromal cells (hMSCs) with multiple myeloma (MM) cells, this

Human Mesenchymal Stem/Stromal Cells

Explaining what a mesenchymal stromal cell (MSC) is, is not such an easy task as one might expect. MSCs are derived from multiple MSCs different sources, serve a wide array of functions and are always isolated as a heterogenous group of cells. This makes it particularly challenging to find a consensus on their exact definition, nomenclature, exact function and *in vivo* differentiation potential. Therefore, the most effective approach to describe hMSCs is to present their historical context.

hMSCs first gained popularity as a stem cell. Stem cells lay the foundation of multicellular organisms. Embryonic stem cells orchestrate the growth and patterning during embryonic development, while adult stem cells are responsible for regeneration during adulthood. The classical definition of a stem cell is that of a relatively undifferentiated cell that divides asymmetrically, producing another stem cell and a differentiated cell (Cooper, 2000; Shenghui et al., 2009). Because of their significance in biology and regenerative medicine, stem cells have become a prominent subject in modern research. Especially human mesenchymal stromal cells (hMSCs) have proven to be a promising candidate in this context (Ullah et al., 2015).

Mesenchyme first appears in embryonic development during gastrulation. There, cells that are committed to a mesodermal fate, lose their cell junctions and exit the epithelial layer in order to migrate freely. This process is called epithelial-mesenchymal transition (Tam & Beddington, 1987; Nowotschin & Hadjantonakis, 2010). Hence, the term mesenchyme describes non-epithelial embryonic tissue differentiating into mesodermal lineages such as bone, muscles and blood. Interestingly, it was shown nearly twenty years earlier that cells within adult bone marrow seemed to have mesenchymal properties as they were able to differentiate into bone tissue (A. J. Friedenstein et al., 1966; A. Friedenstein & Kuralesova, 1971; Bianco, 2014). This was the origin of the "mesengenic process"-hypothesis: This concept states that mesenchymal stem cells serve as progenitors for multiple mesodermal tissues (bone, cartilage, muscle, marrow stroma, tendon, fat, dermis and connective tissue) during both adulthood and embryonic development (A. Caplan, 1991; A. I. Caplan, 1994). The mesenchymal nature of these cells (termed bone marrow stromal cells: BMSCs) was confirmed later when they were shown to differentiate into adipocytic (fat) and chondrocytic (cartilage) lineages (Pittenger et al., 1999). Since then, the term "mesenchymal stem cell" (MSC) has grown popular as an adult multipotent precursor to a couple of mesodermal tissues. hMSCs derived from bone marrow (hMSCs) were shown to differentiate into osteocytes, chondrocytes, adipocytes and cardiomyocytes (Gronthos et al., 1994; Muruganandan et al., 2009; Xu et al., 2004) Most impressively, these cells also exhibited ectodermal and endodermal differentiation potential, as they produced neuronal cells, pancreatic cells and hepatocytes (Barzilay et al., 2009; Wilkins et al., 2009; Gabr et al., 2013; Stock et al., 2014).

Furthermore, cultures with MSC properties can be established from "virtually every postnatal organs and tissues", and not just bone marrow (da Silva Meirelles et al., 2006). However, it has to be noted that hMSCs can differ greatly in their transcription profile and *in vivo* differentiation potential depending on which tissue they originated from (Jansen et al., 2010; Sacchetti et al., 2016).

Since "hMSCs' are a heterogenous group of cells, they were defined by their *in vitro* characteristics. A minimal set of criteria are the following (Dominici et al., 2006): First, hMSCs must be plastic adherent. Second, they must express or lack a set of specific surface antigens (positive for CD73, CD90, CD105; negative for CD45, CD34, CD11b, CD19). Third, hMSCs must differentiate to osteoblasts, adipocytes and chondroblasts *in vitro*. Together, hMSCs exhibit diverse differentiation potentials and can be isolated from multiple sources of the body. This offers great opportunity for regenerative medicine, if the particular hMSC-subtype is properly characterized.

Multiple Myeloma

Multiple myeloma arises from clonal expansion of malignant plasma cells in the bone marrow (BM). At diagnosis, myeloma cells have disseminated to multiple sites in the skeleton and, in some cases, to virtually any tissue (Rajkumar & Kumar, 2020; Bladé et al., 2022).

Myeloma-hMSC Interactions

Since plasma cells can not survive outside the bone marrow, MM cells also require survival signals for growth and disease progression. These signals are produced by the bone marrow microenvironment, including ECM, MSCs and ACs (Kibler et al., 1998; García-Ortiz et al., 2021).

Myeloma Bone Disease

Bone is a two-phase system in which the mineral phase provides the stiffness and the collagen fibers provide the ductility and ability to absorb energy (Viguet-Carrin et al., 2006). On a molecular level, bone tissue is composed of extracellular matrix (ECM) proteins that are calcified by hydroxyapatite crystals. This ECM consists mostly of collagen type I, but also components with major regulatory activity, such as fibronectin and proteoglycans that are essential for healthy bone physiology (Alcorta-Sevillano et al., 2020). Bone tissue is actively remodeled by bone-forming osteoblasts and bone-degrading osteoclasts. Osteoblasts are derived from mes-

enchymal stromal cells (MSCs) that reside in the bone marrow (A. J. Friedenstein et al., 1966; Pittenger et al., 1999). MSCs also give rise to adipocytes (ACs) to form Bone Marrow Adipose Tissue (BMAT), which can account for up to 70% of bone marrow volume (Fazeli et al., 2013).

MM indirectly degrades bone tissue by stimulating osteoclasts and inhibiting osteoblast differentiation, which leads to MM-related bone disease (MBD) (Glavey et al., 2017). MBD is present in 80% of patients at diagnosis and is characterized by osteolytic lesions, osteopenia and pathological fractures (Terpos et al., 2018).

Dissemination of Myeloma Cells

dissemination is still widely unclear - multistep process - invasion, intravasation, intravascular arrest, extravasation, colonization - overcome adhesion, retention, and dependency on the BM microenvironment - loss of adhesion factors such as CD138

Code-Automation as a Standard in Modern Biosciences

Beschreibe die Situation. - Big Data in Biosciences - what is big data, examples - Define citable challenges: - reproducibility crisis - lack of tools

In recent years, the biosciences have evolved dramatically, with a marked increase in the volume and complexity of data generated (Yang et al., 2017; Ekmekci et al., 2016). This transformation necessitates robust software tools, many of which require coding skills to use effectively. Here we summarize standard tools used by biosciences today and show their reliance on coding. The author argues that the role of a modern independent researcher is now intertwined with coding skills similar to a role of "precision medicine bioninformatician" (Gómez-López et al., 2019).

Statistical analysis in biosciences has traditionally been reliant on user-friendly tools like Excel and GraphPad Prism. While Excel by itself is recognized as limited for complex data analysis (Tanavalee et al., 2016; Incerti et al., 2019), GraphPad Prism offers more advanced statistical models .

However, increasingly demands more sophisticated approaches as data sets grow in size and complexity.

R and Python scripts offer more efficient and versatile solutions, enabling complex analyses with a few lines of code (R Core Team, 2018; Vallat, 2018).

Recognizing this trend, Microsoft has integrated a Python interpreter into Excel to computations more accessible within a widely used platform (**?**).

Next-generation sequencing, such as bulk RNAseq, has become affordable, allowing for larger sample sets during a single PhD project. This technology offers advanced tools that are most efficiently used through scripting in R or Python. In the absence of a dedicated statistician, researchers are compelled to learn coding.

In gene ontology, tools such as Metascape facilitate the integration of vast datasets and outputs multiple useful data visualizations. Metascape also provides multiple excel sheets, containing all results, sometimes in a nested format, which provides even further information that's adaptable for specific hypotheses, but given the sheer amount of data, is impractical to analyze manually.

since Metascape returns large Excel sheets with complex nested information, a researcher without coding skills requires manual work to adapt the results to specific research hypotheses.

its true potential is unlocked only when researchers can manipulate and analyze these data through scripting.

Modern gene ontology tools like Metascape offer powerful graphical user interfaces. However, their effectiveness is only possible through standardizing multiple large datasets.

The output from Metascape, large Excel sheets with complex nested information, is more efficiently analyzed through scripting, which is often necessary to adapt metascape results to specific research hypotheses.

Image analysis is another area where coding skills are essential. ImageJ/FIJI, a standard tool in the field, requires scripting for batch processing of multiple images and automating multiple processing steps into a pipeline. While macros can be recorded, understanding the underlying code is necessary for troubleshooting and adapting the macro to new datasets.

In the field of protein structural biology, Pymol is a standard tool that also has a Python command interface.

Similarly, artificial intelligence (AI), a game-changer in biomedicine, primarily uses Python due to its extensive libraries for machine learning and scientific computing. Python is also a standard for integrative biomedicine simulations.

Finally, databases and repositories are essential for storing, retrieving, and sharing data. Researchers need to understand common file formats to adhere to standards that ensure re-usability and interoperability. Scripting helps automate the process of formatting data for submission to these databases.

In conclusion, the integration of coding in bioscience research is not just a trend but a necessity. As the field continues to evolve, the demarcation between biologists and computational scientists blurs, underscoring the importance of coding skills for the next generation of researchers. The ability to code is fast becoming an indispensable asset, as integral to bioscience as traditional laboratory skills.

How Code Quality Improves Scientific Reproducibility

A main reason to write software is to define re-usable instructions for task automation (Narzt et al., 1998). However, the complexity of the code makes it prone to errors and can prevent usage by persons other than the author himself. This is a problem for the general scientific community, as the software is often the only way to reproduce the results of a study (Sandve et al., 2013). Hence, modern journals aim to enforce standards to software development, including software written and used by biological researchers (Smith et al., 2018). Here, we provide a brief overview of the standards utilized by `plotastic` that to ensure its reliability and reproducibility by the scientific community (Peng, 2011).

Modern software development is a long-term commitment of maintaining and improving

code after initial release (Boswell & Foucher, 2011). Hence, it is good practice to write the software such that it is scalable, maintainable and usable. Scalability or, to be precise, structural scalability means that the software can easily be expanded with new features without major modifications to its architecture (Bondi, 2000). This is achieved by writing the software in a modular fashion, where each module is responsible for a single function. Maintainability means that the software can easily be fixed from bugs and adapted to new requirements (Kazman et al., 2020). This is achieved by writing the code in a clear and readable manner, and by writing tests that ensure that the code works as expected (Boswell & Foucher, 2011). Usability is hard to define (Brooke, 1996), yet one can consider a software as usable if the commands have intuitive names and if the software's manual, termed "documentation", is up-to-date and easy to understand for new users with minimal coding experience. A software package that has not received an update for a long time (approx. one year) could be considered abandoned. Abandoned software is unlikely to be fully functional, since it relies on other software (dependencies) that has changed in functionality or introduce bugs that were not expected by the developers of all dependencies. Together, software that's scalable, maintainable and usable requires continuous changes to its codebase. There are best practices that standardize the continuous change of the codebase, including version control, continuous integration (often referred to as CI), and software testing.

Version control is a system that records changes to the codebase line by line, allowing the documentation of the history of the codebase, including who made which changes and when. This is required to isolate new and experimental features into newer versions and away from the stable version that's known to work. The most popular version control system is Git, which is considered the industry standard for software development (Chacon & Straub, 2024). Git can use GitHub.com as a platform to store and host codebases in the form of software repositories. GitHub's most famous feature is called "pull request". A pull request is a request from anyone registered on GitHub to include their changes to the codebase (as in "please pull this into your main code"). One could see pull requests as the identifying feature of the open source community, since it exposes the codebase to potentially thousands of independent developers, reaching a workforce that is impossible to achieve with closed source models used by paid software companies.

Continuous integration (CI) is a software development practice in which developers integrate code changes into a shared repository several times a day (Duvall et al., 2007). Each integration triggers the test suite, aiming to detect errors as soon as possible. The test suite includes building the software, setting up an environment for the software to run and then executing the programmed tests, ensuring that the software runs as a whole. Continuous integration is often used together with software branches. Branches are independent copies of the codebase that are meant to be merged back into the original code once the changes are finished. Since branches

accumulate multiple changes over time, this can lead to minor incompatibilities between the branches of all developers (integration conflicts), which is something that CI helps to prevent.

Continuous integration especially relies on a thorough software testing suite. Software testing is the practice of writing code that checks if the codebase works as expected (Myers et al., 2011). The main type of software testing is unit testing, which tests the smallest units of the codebase (functions and classes) in isolation (Listing 1).

Listing 1: Example of an arbitrary python function and its respective unit test function. The first function simply returns the number 5. The second function tests if the first function indeed returns the number 5. The test function is named with the prefix "`test_`" and is placed in a file that ends with the suffix "`_test.py`". The test function is executed by the testing framework `pytest`. Note that code after "`#`" is considered a comment and won't be executed.

```python
# Define a function called "give_me_five" that returns the number 5
def give_me_five():
    return 5
# Define a test function asserting that "give_me_five" returns 5
def test_give_me_five():
    assert give_me_five() == 5
```

The quality of the software testing suite is measured by the code coverage, the precision of the tests, and the number of test-cases that are checked. The code coverage is the percentage of the codebase that is called by the testing functions, which should be as close to 100% as possible, although it does not measure how well the code is tested. The precision of the test is not a measurable quantity, but it represents if the tests truly checks if the code works as expected. The number of test-cases is the number of different scenarios that are checked by the testing functions, for example testing every possible option or combinations of options for functions that have multiple options. The most popular software testing framework for python is `pytest`, which is utilized by `plotastic` (Krekel et al., 2004).

Together, the standards of software development, including version control, continuous integration, and software testing, ensure that the software is scalable, maintainable, and usable. This is especially important for software that is used by the scientific community, as it ensures that the software is working as expected at defined versions years after publishing scientific results.

Python as a Programming Language

Here, we provide a general overview of the python programming language, explaining terms like *"type"*, *"method"*, etc., in order to prepare readers without prior programming experience for the following chapters. We also describe the design principles of python to lay out the key concepts that differentiate python compared to other programming languages. A more detailed tutorial on python that's specialized for bioscientists is found in Ekmekci et al. 2016

Listing 2: Example of readable python code. This one-line code returns the words (string) `'Hello, World!'` when executed. The command is straightforward and easy to understand.

```python
print("Hello, World!")
# Output: Hello, World!
```

Languages such as python are considered *"high-level"*, which means that it is designed to be easy to read and write, but also independent of hardware by hiding (*"abstracting"*) underlying details (*The Python Language Reference*, n.d.). A key principle of python is the emphasis on implementing a syntax that is concise and close to human language (Listing 2, Listing 3).

Listing 3: Example of less readable code written in the low-level programming language C. This code is doing exactly the same as the python code in Listing 2. The command is harder to understand because more steps are needed to access the same functionality, including the definition of a function

```c
#include <stdio.h>
int main() {
    printf("Hello, World!");
    return 0;
}
// Output: Hello, World!
```

Furthermore, python is an *interpreted* language, which means that the code is executed line by line. This makes coding easier because the programmer can see the results of the code immediately after writing it, and error messages point to the exact line where the error occurred. This is in contrast to *compiled* languages, where the code has to be compiled into machine code before it can be executed. The advantage of compiled languages is that the code runs faster, because the machine code is optimized for the hardware.

Python automates tasks that would otherwise require an advanced understanding of computer hardware, like the need for manual allocation of memory space. This is achieved by using a technique called *"garbage collection"*, which automatically frees memory space that is no longer needed by the program. This is a feature that is not present in low-level programming languages like C or C++, that were designed to maximize control over hardware.

Another hallmark of python is its *dynamic typing system*. In python the type is inferred automatically during code execution (Listing 4). This is in contrast to *statically* typed languages like C, where the type of a variable has to be declared explicitly and cannot be changed during code execution (Listing 5) (*The Python Language Reference*, n.d.).

Dynamic typing makes python a very beginner-friendly language, since one does not have to keep track of the type of each variable. However, this also makes python a slower language, because the interpreter has to check the type of each variable during code execution. Also, developing code with dynamic typing systems is prone to introducing bugs ("type errors"), because it allows unexperienced developers to convert variables from one type to another without noticing, leading to unexpected behavior. Hence, larger python projects require disciplined

Listing 4: Example of dynamic typing in python. The variable "a" is assigned the value 5, which is of type integer. The variable "a" is then assigned the value "Hello, World!", which is of type string. Python allows dynamic re-assignment of variables with different types. Note that code after "#" is considered a comment and won't be executed.

```
a = 5  # Type integer
a = 5.0  # Type float
a = 'Hello, World!'  # Type string
a = True  # Type boolean
a = False  # Type boolean
a = [1, 2, 3]  # Type list of integers
a = {'name': 'Regina'}  # Type dictionary
```

Listing 5: Example of static typing in C. The variable "a" is declared as an integer (`int`), and can only store integers. The variable "a" is then assigned the value 5, which is an integer. The variable "a" is then assigned the value `'Hello, World!'`, which is a string. This results in a compilation error, because the variable "a" can only store integers. Note that code after "//" is considered a comment and won't be executed.

```
int a;  // Declare type as integer
a = 5;
a = 'Hello, World!';  // Compilation error!
```

adherence to programming conventions. One such convention is *type hinting*, which is a way to explicitly note the type of a variable. Type hinting does not have an effect on the code, but it makes the code more readable and understandable for other developers, and allows for development environments to detect type errors before execution (Listing 6) (van Rossum et al., 2014).

Listing 6: Example of type hints used in python. Explicitly stating the type of the variable is optional and does not change the behavior of the code as shown in Listing 4.

```
a: int = 5
a: str = 'Hello, World!'
```

Python supports both functional and object-oriented programming paradigms. In functional programming, the code is written in a way that the program is a sequence of function calls, where each function call returns a value that is used in the next function call (Listing 7). This approach is useful when multiple actions have to be performed on the same data and the structure of the data is relatively simple, for example a string of a gene sequence.

When the data itself gains in complexity, for example when storing not just the gene sequence, but also the promotor sequence, an object-oriented approach is more suitable (Listing 8). Object-oriented programming is a programming paradigm that uses objects and classes. An object is a collection of both data and functions, and a class is a blueprint for creating objects. The data of an object is stored as attributes. Functions that are associated with an object are called methods.

Listing 7: Example of functional programming in Python. The code defines a function called "`find_restriction_site`" that finds the position of a restriction site in a gene. The function "`cut`" uses the function "`find_restriction_site`" to cut the gene at the restriction site.

```python
def find_restriction_site(gene: str):
    return gene.find('GCGC')

def cut(gene: str):
    position = find_restriction_site(gene)
    return gene[position:]

gene1 = 'TGAGCTGAGCTGATGCGCTATATTTAGGCG'
gene1_cut = cut(gene1)
print(gene1_cut)
# Output: GCGCTATATTTAGGCG
```

Listing 8: Example of object oriented programming in python. The class is called "`Gene`" and has four methods, "`__init__`", "`find_promotor`", "`find_restriction_site`" and "`cut`". The method "`__init__`" is called when creating ("initializing") an object, which fills the object with user-defined data. The parameter "`self`" is used to reference the object itself internally. "`find_promotor`" is a method that finds the position of the promotor in the gene and is called during object initialization.

```python
class Gene:
    def __init__(self, sequence: str):
        self.sequence: str = sequence # Save sequence as attribute
        self.promotor: str = self.find_promotor()
    def find_promotor(self):
        return self.sequence.find('TATA')
    def find_restriction_site(self):
        return self.sequence.find('GCGC')
    def cut(self):
        position = self.find_restriction_site()
        return self.sequence[position:]

gene1 = Gene(sequence='TGAGCTGAGCTGATGCGCTATATTTAGGCG') # Create object
gene1_cut = gene1.cut() # Call the method cut
print(gene1_cut) # Show result
# Output: GCGCTATATTTAGGCG
```

A major benefit of using an object oriented versus a functional approach is that the data itself programmable, enabling the programmer to define the behavior of the data itself through methods. This is achieved by using the keyword "`self`" to reference the object itself inside the class. For example, one could extend the class "`Gene`" with a method that finds the promotor of the gene and stores it as an attribute (Listing 8).

When designing software, both functional and object oriented programming can be used together, where object oriented programming is often used to design the program's overall architecture, and functional programming is used to implement the algorithms of the program's features. This allows for scalability of the software, as every single class is extended through the

addition of new methods. Furthermore, classes can be expanded in their functionalities through inheritance (Listing 9).

Listing 9: Example of inheritance in python. The class "mRNA" inherits from the class "Gene". The class "mRNA" has two methods, "__init__" and "find_stopcodon". The method "find_stopcodon" finds the position of stop codons.

```python
# Define a class called mRNA inheriting from the class Gene
class mRNA(Gene):
    def __init__(self, sequence: str):
        super().__init__(sequence)  # Get attributes from parent class
        self.sequence.replace('T', 'U')  # Replace thymine with uracil
    def find_stopcodons(self):
        return self.sequence.find('UGA')

mrna1 = mRNA(sequence='TGAGCTGAGCTGATGCGCTATATTTAGGCG') # Create object
print(mrna1.find_stopcodons())  # Call the method translate
# Output: [0, 5, 10]
```

Inheritance is a feature of object-oriented programming that allows a class to access every attribute and method of a parent class. For example, one could extend the class "Gene" with a class "mRNA", by writing a class "mRNA" that inherits from the class "Gene".

Together, python is not just beginner-friendly, but also well respected for its ease in development, which is why it is widely used in professional settings for web development, data analysis, machine learning, biosciences and more (Ekmekci et al., 2016).

Data Science with Python

the ease of use has made python a very popular language (Rayhan & Gross, 2023)

Like any other programming language, python alone does not provide specialized tools like those used for data analysis (*The Python Language Reference*, n.d.). However, python was designed to be extended by packages developed by its users. A python package consists of multiple python modules, where each module is a text-file with a `.py` ending containing python code. Famous examples of such packages are `pytorch` and `tensorflow`, that are used to build models of artificial intelligence, including ChatGPT (Paszke et al., 2019; Abadi et al., 2016; Radford et al., 2019). Here, we outlay the most important packages used for `plotastic`.

Interactive Python - Jupyter

Python overcame the issues of interpreted language by utilizing Code written in C numpy: - Acceleration, - SIMD instructions

Tabular operations - pandas

Data visualization - matplotlib - seaborn

Inferential Statistics - pingouin

AI: - pytorch and tensorflow - example: VGG19 is just a few lines of code (**??**) asdfdf

Aims

This project defines these aims:

- Characterise the interaction between myeloma cells and mesenchymal stromal cells
- Aim 2
- Aim 3

Chapter 1: Modelling Myeloma Dissemination *in vitro*

Abstract

lorem ipsum

Introduction

lorem ipsum dolor sit amet

Methods

lorem ipsum

Results

lorem ipsum

Discussion

lorem ipsum

Chapter 2: Semi-Automating Data Analysis with plotastic

Abstract

plotastic addresses the challenges of transitioning from exploratory data analysis to hypothesis testing in Python's data science ecosystem. Bridging the gap between seaborn and pingouin, this library offers a unified environment for plotting and statistical analysis. It simplifies the workflow with user-friendly syntax and seamless integration with familiar seaborn parameters (y, x, hue, row, col). Inspired by seaborn's consistency, plotastic utilizes a DataAnalysis object to intelligently pass parameters to pingouin statistical functions. Hence, statistics and plotting are performed on the same set of parameters, so that the strength of seaborn in visualizing multidimensional data is extended onto statistical analysis. In essence, plotastic translates seaborn parameters into statistical terms, configures statistical protocols based on intuitive plotting syntax and returns a matplotlib figure with known customization options and more. This approach streamlines data analysis, allowing researchers to focus on correct statistical testing and less about specific syntax and implementations.

Introduction

The reproducibility crisis in research highlights a significant challenge in contemporary biosciences, where a substantial portion of studies faces reproducibility issues (Begley & Ioannidis, 2015). One critical yet often overlooked aspect contributing to this crisis is data management. The literature most often refers to *big data* as the main challenge (Gomez-Cabrero et al., 2014). However, these challenges are also present in smaller datasets, which the author refers to as *semi-big data*. This term describes datasets that, while not extensive enough to necessitate advanced computational tools typically reserved for *big data*, are sufficiently large to render manual analysis very time-intensive. *Semi-big data* is often generated by methods like automated microscopy or multiplex qPCR, which produce volumes of data that are manageable on a surface level, but pose substantial barriers for in-depth, manual reproducibility (Bustin, 2014). This is further complicated by the complexity inherent in multidimensional datasets. For example, the qPCR experiment from Chapter 1, Fig. 4 involves the analysis of 19 genes across in three subpopulations, including eleven biological and three technical replicates, resulting in a total of 1881 data points that are all assigned to a complex set of experimental variables. Without a clearly documented data analysis protocol and standardized data formats, the reproduction of such analysis becomes extremely challenging, if not impossible (Bustin, 2014).

The evolving standards in data analysis advocate for the standardization of analytical pipelines, rationalization of sample sizes, and enhanced infrastructure for data storage, address-

ing some of these challenges (Goodman et al., 2016; Wilkinson et al., 2016). However, these advancements can place undue pressure on researchers, particularly those with limited training in statistics, underscoring the need for intuitive, user-friendly analytical tools (Gosselin, 2021; Armstrong, 2014; Gómez-López et al., 2019)

In this context, plotastic emerges as a tool designed to democratize access to sophisticated statistical analysis, offering a user-centric interface that caters to researchers across varying levels of statistical proficiency. By integrating robust statistical methodologies within an accessible framework, plotastic aims to contribute to enhancing the reproducibility and integrity of research in the biosciences (Gomez-Cabrero et al., 2014).

initially, the need to develop plotastic arose during this project. The first is to address the author's need for a tool that could handle the complex, multidimensional data generated by e.g. qPCR experiments. These experiments typically involve the analysis of multiple genes across several time points and biological replicates, resulting in datasets that are challenging to analyze manually. The author's experience with traditional statistical software, such as Prism, revealed that these tools required extensive manual input, making them unsuitable for the efficient analysis of complex, multidimensional data. - The second was to increase speed. THis is required for developing methods

Since plotastic optimizes the analysis of *semi-big data*, we introduce the term *semi-automation* to distinguish itself from the fully automated pipelines used for *big data*. Semi-automation is defined as the following aspects:

1. **Semi-big input:** The input size is oriented towards *semi-big data*, which is characterized as being manageable by manual analysis, yet highly time inefficient, and probably impossible to re-analyse by someone else than the researcher.

2. **Standardized input** The input follows a standardized format (e.g. long-format)

3. **Minimize user configuration:** User configuration is strictly minimized. The user is never asked to pass the same parameters twice. This reduces the risk of human error and time spent on configuration.

4. **Default configuration provides acceptable results:** If the user does not provide any manual configuration, the pipeline should provide acceptable results. Options should be provided to allow a level of flexibility to adapt the pipeline to the user's needs.

5. **Small Reviewable Processing Steps:** The analysis steps are structured into small processes that can be combined to form a complete analysis pipeline. That way, each step can act as a stage for quality control to improve error detection and troubleshooting. For

a statistical analysis, that means the processing steps are separated into 3 steps, those being assumption testing, factor analysis and post-hoc testing.

6. **Isolated Steps:** Processing steps should work independently from another, in the best case only depending on the raw data input. If a processing step depends on the output from other steps, the software should tell the user what exact steps it expects.

7. **Human readable outputs:** Every processing step may provide an output that is not necessarily standardized, but is required to be human readable to ensure reviewability.

Challenges: - Reproducibility crisis? - Data is exploding - Demands for rigorous statistical analysis are increasing - Biologists are not trained in statistics

The demands are rising: (Moreno-Indias et al., 2021)

As laid out in the introduction, one can doubt if a PhD student without coding skillsis at its max efficiency.

Why does Biomedicine need plotastic?: - Thorough analysis has become a standard, with assumption testing, omnibus tests and post-hoc analyses for every experiment. - But data is increasing - Example of my data? - The number of dedicated statisticians is limited - The know-how of statistics in biology is limited, for example, Some authors ignored the problem of multiple testing while others used the method uncritically with no rationale or discussion (Perneger, 1998; Armstrong, 2014)

Why did I need plotastic?

Why do biologists need plotastic? - Assays output more data in shorter time, e.g. multiplex qPCR - example: 20 genes, 3 timepoints, 11 biological replicates, (all 3 technical replicates already averaged) - 20 * 3 * 11 = 660 data points

this is multidimensional data: 660 data points spread across two dimensions: time and gene

-in manual analysis e.g. in Excel, the user has to manually select the data, copy it, paste it into a new sheet, and then perform the statistical test. In Prism, the user has to select the data, click on the statistical test, and then select the data again. This is not only time-consuming, but also prone to

- Re-Analysis: The user has to repeat the process for every gene and timepoint. This is not only time-consuming, but also prone to errors.

shortly Describe Main Packages in more detail: - seaborn: It multidimensional data - pingouin: It's a statistical package

Statement of Need

Python's data science ecosystem provides powerful tools for both visualization and statistical testing. However, the transition from exploratory data analysis to hypothesis testing can be cumbersome, requiring users to switch between libraries and adapt to different syntaxes. seaborn has become a popular choice for plotting in Python, offering an intuitive interface. Its statistical functionality focuses on descriptive plots and bootstrapped confidence intervals (Waskom, 2021). The library pingouin offers an extensive set of statistical tests, but it lacks integration with common plotting capabilities (Vallat, 2018). statannotations integrates statistical testing with plot annotations, but uses a complex interface and is limited to pairwise comparisons (Charlier et al., 2022).

plotastic addresses this gap by offering a unified environment for plotting and statistical analysis. With an emphasis on user-friendly syntax and integration of familiar seaborn parameters, it simplifies the process for users already comfortable with seaborn. The library ensures a smooth workflow, from data import to hypothesis testing and visualization.

Example

The following code demonstrates how plotastic analyzes the example dataset "fmri", similar to Waskom (2021) (Figure 1).

```
### IMPORT PLOTASTIC
import plotastic as plst

# IMPORT EXAMPLE DATA
DF, _dims = plst.load_dataset("fmri", verbose = False)

# EXPLICITLY DEFINE DIMENSIONS TO FACET BY
dims = dict(
      y = "signal",    # y-axis, dependent variable
      x = "timepoint", # x-axis, independent variable (within-subject factor)
      hue = "event",   # color,  independent variable (within-subject factor)
      col = "region"   # axes,   grouping variable
)
# INITIALIZE DATAANALYSIS OBJECT
DA = plst.DataAnalysis(
      data=DF,           # Dataframe, long format
      dims=dims,         # Dictionary with y, x, hue, col, row
      subject="subject", # Datapoints are paired by subject (optional)
      verbose=False,     # Print out info about the Data (optional)
)
# STATISTICAL TESTS
DA.check_normality()   # Check Normality
DA.check_sphericity()  # Check Sphericity
```

```
DA.omnibus_rm_anova()  # Perform RM-ANOVA
DA.test_pairwise()     # Perform Posthoc Analysis

# PLOTTING
(DA
.plot_box_strip()     # Pre-built plotting function initializes plot
.annotate_pairwise(   # Annotate results from DA.test_pairwise()
      include="__HUE" # Use only significant pairs across each hue
      )
)
```

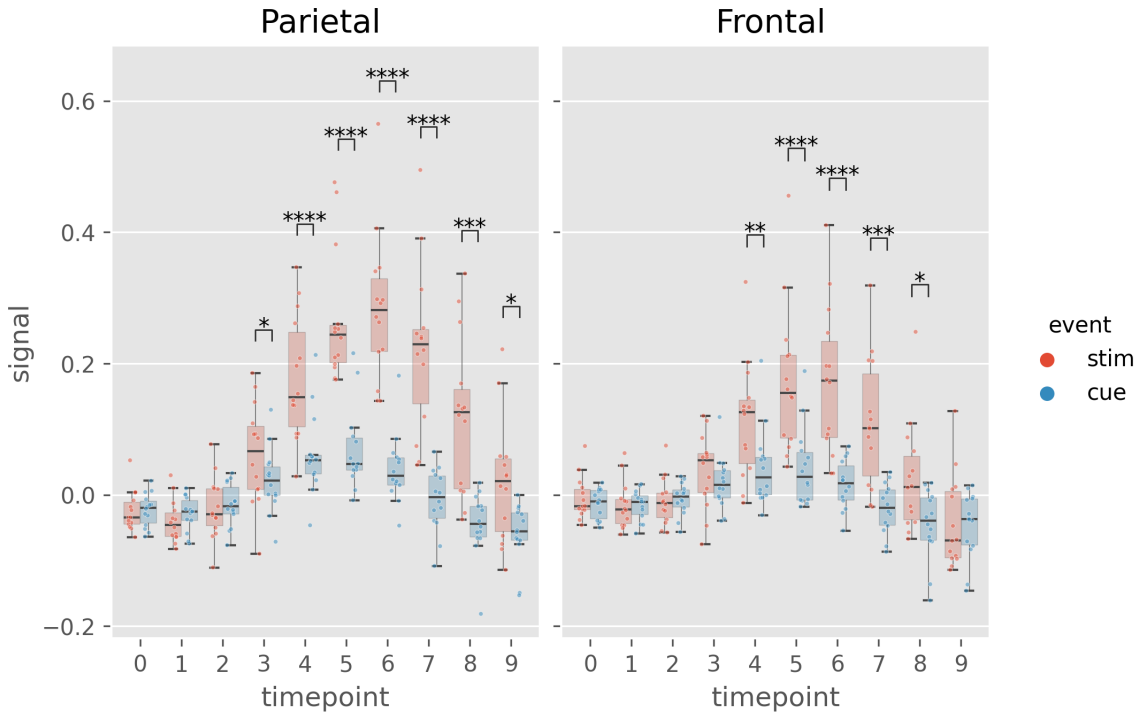


Figure 1: Example figure of plotastic (version 0.1). Image style was set by plt.style.use('ggplot')

Table 1: Results from DA.check_sphericity(). plotastic assesses sphericity after grouping the data by all grouping dimensions (hue, row, col). For example, DA.check_sphericity() grouped the 'fmri' dataset by "region" (col) and "event" (hue), performing four subsequent sphericity tests for four datasets.

'region', 'event'	spher	W	chi2	dof	pval	group count	n per group
'frontal', 'cue'	True	3.26e+20	-462.7	44	1	10	[14]
'frontal', 'stim'	True	2.45e+17	-392.2	44	1	10	[14]
'parietal', 'cue'	True	1.20e+20	-452.9	44	1	10	[14]
'parietal', 'stim'	True	2.44e+13	-301.9	44	1	10	[14]

Table 2: Results of DA.omnibus_rm_anova(). plotastic performs one two-factor RM-ANOVA per axis (grouping the data by row and col dimensions) using x and hue as the within-factors. For this example, DA.omnibus_rm_anova() grouped the 'fmri' dataset by "region" (col), performing two subsequent two-factor RM-ANOVAs. Within-factors are "timepoint" (x) and "event" (hue). For conciseness, GG-Correction and effect sizes are not shown.

'region'	Source	SS	ddof1	ddof2	MS	F	p-unc	stars
'parietal'	timepoint	1.583	9	117	0.175	26.20	3.40e-24	****
'parietal'	event	0.770	1	13	0.770	85.31	4.48e-07	****
'parietal'	timepoint * event	0.623	9	117	0.069	29.54	3.26e-26	****
'frontal'	timepoint	0.686	9	117	0.076	15.98	8.28e-17	****
'frontal'	event	0.240	1	13	0.240	23.44	3.21e-4	***
'frontal'	timepoint * event	0.242	9	117	0.026	13.031	3.23e-14	****

Overview

The functionality of plotastic revolves around a seamless integration of statistical analysis and plotting, leveraging the capabilities of pingouin, seaborn, matplotlib and statannotations (Vallat, 2018; Waskom, 2021; Hunter, 2007; Charlier et al., 2022). It utilizes long-format pandas DataFrames as its primary input, aligning with the conventions of seaborn and ensuring compatibility with existing data structures (Wickham, 2014; Team, 2020; McKinney, 2010).

plotastic was inspired by seaborn using the same set of intuitive and consistent parameters (y, x, hue, row, col) found in each of its plotting functions (Waskom, 2021). These parameters intuitively delineate the data dimensions plotted, yielding 'facetted' subplots, each presenting y against x. This allows for rapid and insightful exploration of multidimensional relationships. plotastic extends this principle to statistical analysis by storing these seaborn parameters (referred to as dimensions) in a DataAnalysis object and intelligently passing them to statistical functions of the pingouin library. This approach is based on the impression that most decisions during statistical analysis can be derived from how the user decides to arrange the data in a plot. This approach also prevents code repetition and streamlines statistical analysis. For example, the subject keyword is specified only once during DataAnalysis initialisation, and plotastic selects the appropriate paired or unpaired version of the test. Using pingouin alone requires the user to manually pick the correct test and to repeatedly specify the subject keyword in each testing function.

In essence, plotastic translates plotting parameters into their statistical counterparts. This translation minimizes user input and also ensures a coherent and logical connection between plotting and statistical analysis. The goal is to allow the user to focus on choosing the correct statistical test (e.g. parametric vs. non-parametric) and worry less about specific implementations.

At its core, plotastic employs iterators to systematically group data based on various

dimensions, aligning the analysis with the distinct requirements of tests and plots. Normality testing is performed on each individual sample, which is achieved by splitting the data by all grouping dimensions and also the x-axis (hue, row, col, x). Sphericity and homoscedasticity testing is performed on a complete sampleset listed on the x-axis, which is achieved by splitting the data by all grouping dimensions (hue, row, col) (Table 1). For omnibus and posthoc analyses, data is grouped by the row and col dimensions in parallel to the matplotlib axes, before performing one two-factor analysis per axis using x and hue as the within/between-factors. (Table 2).

DataAnalysis visualizes data through predefined plotting functions designed for drawing multi-layered plots. A notable emphasis within plotastic is placed on showcasing individual datapoints alongside aggregated means or medians. In detail, each plotting function initializes a matplotlib figure and axes using plt.subplots() while returning a DataAnalysis object for method chaining. Axes are populated by seaborn plotting functions (e.g., sns.boxplot()), leveraging automated aggregation and error bar displays. Keyword arguments are passed to these seaborn functions, ensuring the same degree of customization. Users can further customize plots by chaining DataAnalysis methods or by applying common matplotlib code to override plotastic settings. Figures are exported using plt.savefig().

plotastic also focuses on annotating statistical information within plots, seamlessly incorporating p-values from pairwise comparisons using statannotations (Charlier et al., 2022). This integration simplifies the interface and enables options for pair selection in multidimensional plots, enhancing both user experience and interpretability.

For statistics, plotastic integrates with the pingouin library to support classical assumption and hypothesis testing, covering parametric/non-parametric and paired/non-paired variants. Assumptions such as normality, homoscedasticity, and sphericity are tested. Omnibus tests include two-factor RM-ANOVA, ANOVA, Friedman, and Kruskal-Wallis. Posthoc tests are implemented through pingouin.pairwise_tests(), offering (paired) t-tests, Wilcoxon, and Mann-Whitney-U.

To sum up, plotastic stands as a unified and user-friendly solution catering to the needs of researchers and data scientists, seamlessly integrating statistical analysis with the power of plotting in Python. It streamlines the workflow, translates seaborn parameters into statistical terms, and supports extensive customization options for both analysis and visualization.

Discussion

Is plotastic tested? Coverage? Does it cover every feature? What is not covered

The Architecture of plotastic is shown in Appendix A Figure 3.

The full code of an example analysis is shown in subsection C.3.

Is plotastic USABLE for biologists? - Yes but use is limited by minimal knowledge of Python - However, that is subject to change as Python is becoming more popular in biology and AI assisted coding decreased the barrier to entry significantly. Tools like github copilot are able to generate code, fix bugs and suggest improvements. This is a game changer for biologists that are not familiar with programming. - Furthermore, installing and using plotastic for biologists is overestimated. These steps re needed: - Install anaconda from the internet - Open the terminal - Type `pip install plotastic` - Check Rea

The evaluation of plotastic within this thesis reflects its potential to address key challenges in the field of data analysis. The software integrates a comprehensive suite of statistical tests, such as ANOVA and t-tests, designed for adaptability and ease of use, leveraging the functionalities of pingouin.

In the context of the reproducibility crisis in scientific research, plotastic offers noteworthy contributions, though it is not positioned as a universal remedy. The tool's unique approach to integrating statistical analysis with visual representation establishes a new paradigm, promoting methodological transparency. By mandating that statistical analyses accompany relevant graphical outputs, plotastic ensures that analyses are not only conducted with proper scientific rigor but also documented in a manner that facilitates replication, provided the user possesses proficiency in Python.

Usability is a critical attribute of analytical software, particularly as researchers confront increasingly complex datasets. While the developer's intimate familiarity with plotastic may bias perceptions of its ease of use, it is recognized that novices may initially encounter challenges. Nevertheless, plotastic is distinguished by its user-friendly interface, enabling users with minimal statistical training to perform sophisticated analyses by intuitively mapping plotting concepts to statistical operations.

The transition to a new analytical framework, especially one that incorporates coding, presents a learning curve. However, the advantages of plotastic in terms of analytical clarity, speed, and depth are anticipated to outweigh these initial challenges. Support mechanisms, such as assistance from advanced AI like ChatGPT, are available to mitigate these hurdles, supporting users across varying levels of expertise.

In conclusion, plotastic is posited as a valuable tool in the landscape of scientific research, offering a means to enhance the reproducibility and efficiency of data analysis. Its development ethos emphasizes simplifying complex analytical tasks, thereby contributing to the broader goal of fostering transparent and reproducible research practices.

DO we apply the principles of Semi-Automation to the software?

what features are missing? - Bivariate analysis - Filer to help save the output? - StatResults: System to suggest the correct test, based on the data

Summarising Discussion

Time Lapse

Lorem Ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum

Myeloma

Lorem Ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum

Semi-Automated Analysis Improves Agility During Establishing new *in vitro* Methods

Was plotastic useful for me? - Yes incredibly. I was able to perform the statistical tests and visualize the data in a fraction of the time that I would have needed manually. This allowed me to focus on the interpretation of the results and the writing of the manuscript. There was one particular example where my analysis was so fast, that I fed raw datatables during microscopy into python scripts and was able to adapt the experimental technique during the experiment. This allows for an agile and adaptive work environment that is not possible with manual analysis and proved invaluable during development of *in vitro* methods. - These experiments benefited from the use of plotastic, as the

Further research is needed to assess the true impact of semi-automated analysis on the agility

of establishing new *in vitro* methods.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., … Zheng, X. (2016, March). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems* (No. arXiv:1603.04467). arXiv. Retrieved 2024-03-07, from http://arxiv.org/abs/1603.04467 doi: 10.48550/arXiv.1603.04467

Alcorta-Sevillano, N., Macías, I., Infante, A., & Rodríguez, C. I. (2020, December). Deciphering the Relevance of Bone ECM Signaling. *Cells*, *9*(12), 2630. Retrieved 2023-12-20, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7762413/ doi: 10.3390/cells9122630

Armstrong, R. A. (2014, September). When to use the Bonferroni correction. *Ophthalmic & Physiological Optics: The Journal of the British College of Ophthalmic Opticians (Optometrists)*, *34*(5), 502–508. doi: 10.1111/opo.12131

Barzilay, R., Ben-Zur, T., Bulvik, S., Melamed, E., & Offen, D. (2009, May). Lentiviral delivery of LMX1a enhances dopaminergic phenotype in differentiated human bone marrow mesenchymal stem cells. *Stem cells and development*, *18*(4), 591–601. doi: 10.1089/scd.2008.0138

Begley, C. G., & Ioannidis, J. P. A. (2015, January). Reproducibility in science: Improving the standard for basic and preclinical research. *Circulation Research*, *116*(1), 116–126. doi: 10.1161/CIRCRESAHA.114.303819

Bianco, P. (2014). "Mesenchymal" stem cells. *Annual review of cell and developmental biology*, *30*, 677–704. doi: 10.1146/annurev-cellbio-100913-013132

Bladé, J., Beksac, M., Caers, J., Jurczyszyn, A., von Lilienfeld-Toal, M., Moreau, P., … Richardson, P. (2022, March). Extramedullary disease in multiple myeloma: A systematic literature review. *Blood Cancer Journal*, *12*(3), 1–10. Retrieved 2023-03-24, from https://www.nature.com/articles/s41408-022-00643-3 doi: 10.1038/s41408-022-00643-3

Bondi, A. B. (2000, September). Characteristics of scalability and their impact on performance. In *Proceedings of the 2nd international workshop on Software and performance* (pp. 195–203). New York, NY, USA: Association for Computing Machinery. Retrieved 2024-03-07, from https://dl.acm.org/doi/10.1145/350391.350432 doi: 10.1145/350391.350432

Boswell, D., & Foucher, T. (2011). *The Art of Readable Code: Simple and Practical Techniques for Writing Better Code*. "O'Reilly Media, Inc.".

Brooke, J. (1996, January). SUS – a quick and dirty usability scale. In (pp. 189–194).

Bustin, S. A. (2014, December). The reproducibility of biomedical research: Sleepers awake! *Biomolecular Detection and Quantification*, *2*, 35–42. Retrieved 2024-03-18, from https://www.sciencedirect.com/science/article/pii/S2214753515000030 doi: 10.1016/j.bdq.2015.01.002

Caplan, A. (1991). Mesenchymal stem cells. *Journal of orthopaedic research : official publication of the Orthopaedic Research Society*, *9*(5), 641–50. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/1870029 doi: 10.1002/jor.1100090504

Caplan, A. I. (1994, July). The mesengenic process. *Clinics in plastic surgery*, *21*(3), 429–435.

Chacon, S., & Straub, B. (2024, March). *Git - Book*. Retrieved 2024-03-07, from https://git-scm.com/book/de/v2

Charlier, F., Weber, M., Izak, D., Harkin, E., Magnus, M., Lalli, J., … Repplinger, S. (2022, October). *Trevismd/statannotations: V0.5*. Zenodo. Retrieved 2023-11-16, from https://zenodo.org/record/7213391 doi: 10.5281/ZENODO.7213391

Cooper, G. M. (2000). The Cell: A Molecular Approach. 2nd Edition. *Sinauer Associates*, Proliferation in Development and Differentiation. Retrieved from https://www.ncbi.nlm.nih.gov/books/NBK9906/

da Silva Meirelles, L., Chagastelles, P. C., & Nardi, N. B. (2006, June). Mesenchymal stem cells reside in virtually all post-natal organs and tissues. *Journal of cell science*, *119*(Pt 11), 2204–2213. doi: 10.1242/jcs.02932

Dominici, M., Le Blanc, K., Mueller, I., Slaper-Cortenbach, I., Marini, F., Krause, D., … Horwitz, E. (2006). Minimal criteria for defining multipotent mesenchymal stromal cells. The International Society for Cellular Therapy position statement. *Cytotherapy*, *8*(4), 315–317. doi: 10.1080/14653240600855905

Duvall, P., Matyas, S., & Glover, A. (2007). *Continuous integration: Improving software quality and reducing risk*. Pearson Education. Retrieved from https://books.google.de/books?id=PV9qfEdv9L0C

Ekmekci, B., McAnany, C. E., & Mura, C. (2016, July). An Introduction to Programming for Bioscientists: A Python-Based Primer. *PLOS Computational Biology*, *12*(6), e1004867. Retrieved 2024-03-10, from https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004867 doi: 10.1371/journal.pcbi.1004867

Fazeli, P. K., Horowitz, M. C., MacDougald, O. A., Scheller, E. L., Rodeheffer, M. S., Rosen, C. J., & Klibanski, A. (2013, March). Marrow Fat and Bone—New Perspectives. *The Journal of Clinical Endocrinology and Metabolism*, *98*(3), 935–945. Retrieved 2023-12-20, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3590487/ doi: 10.1210/jc.2012-3634

Friedenstein, A., & Kuralesova, A. I. (1971, August). Osteogenic precursor cells of bone marrow in radiation chimeras. *Transplantation*, *12*(2), 99–108.

Friedenstein, A. J., Piatetzky-Shapiro, I. I., & Petrakova, K. V. (1966, December). Osteogenesis in transplants of bone marrow cells. *Journal of embryology and experimental morphology*, *16*(3), 381–390.

Gabr, M. M., Zakaria, M. M., Refaie, A. F., Ismail, A. M., Abou-El-Mahasen, M. A., Ashamallah, S. A., … Ghoneim, M. A. (2013). Insulin-producing cells from adult human bone marrow mesenchymal stem cells control streptozotocin-induced diabetes in nude mice. *Cell transplantation*, *22*(1), 133–145. doi: 10.3727/096368912X647162

García-Ortiz, A., Rodríguez-García, Y., Encinas, J., Maroto-Martín, E., Castellano, E., Teixidó, J., & Martínez-López, J. (2021, January). The Role of Tumor Microenvironment in Multiple Myeloma Development and Progression. *Cancers*, *13*(2). Retrieved 2021-02-02, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7827690/ doi: 10.3390/cancers13020217

Glavey, S. V., Naba, A., Manier, S., Clauser, K., Tahri, S., Park, J., … Ghobrial, I. M. (2017, November). Proteomic characterization of human multiple myeloma bone marrow extracellular matrix. *Leukemia*, *31*(11), 2426–2434. Retrieved 2023-09-05, from https://www.nature.com/articles/leu2017102 doi: 10.1038/leu.2017.102

Gomez-Cabrero, D., Abugessaisa, I., Maier, D., Teschendorff, A., Merkenschlager, M., Gisel, A., … Tegnér, J. (2014, March). Data integration in the era of omics: Current and future challenges. *BMC Systems Biology*, *8*(2), I1. Retrieved 2024-03-18, from https://doi.org/10.1186/1752-0509-8-S2-I1 doi: 10.1186/1752-0509-8-S2-I1

Gómez-López, G., Dopazo, J., Cigudosa, J. C., Valencia, A., & Al-Shahrour, F. (2019, May). Precision medicine needs pioneering clinical bioinformaticians. *Briefings in Bioinformatics*, *20*(3), 752–766. doi: 10.1093/bib/bbx144

Goodman, S. N., Fanelli, D., & Ioannidis, J. P. A. (2016, June). What does research reproducibility mean? *Science Translational Medicine*, *8*(341), 341ps12-341ps12. Retrieved 2024-03-18, from https://www.science.org/doi/10.1126/scitranslmed.aaf5027 doi: 10.1126/scitranslmed.aaf5027

Gosselin, R.-D. (2021, February). Insufficient transparency of statistical reporting in preclinical research: A scoping review. *Scientific Reports*, *11*(1), 3335. Retrieved 2024-03-11, from https://www.nature.com/articles/s41598-021-83006-5 doi: 10.1038/s41598-021-83006-5

Gronthos, S., Graves, S. E., Ohta, S., & Simmons, P. J. (1994, December). The STRO-1+ fraction of adult human bone marrow contains the osteogenic precursors. *Blood*, *84*(12), 4164–4173.

Hunter, J. D. (2007, May). Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*,

9(3), 90–95. Retrieved 2023-11-15, from https://ieeexplore.ieee.org/document/4160265 doi: 10.1109/MCSE.2007.55

Incerti, D., Thom, H., Baio, G., & Jansen, J. P. (2019, May). R You Still Using Excel? The Advantages of Modern Software Tools for Health Technology Assessment. *Value in Health*, *22*(5), 575–579. Retrieved 2024-03-11, from https://www.sciencedirect.com/science/article/pii/S1098301519300506 doi: 10.1016/j.jval.2019.01.003

Jansen, B. J. H., Gilissen, C., Roelofs, H., Schaap-Oziemlak, A., Veltman, J. A., Raymakers, R. A. P., … Adema, G. J. (2010, April). Functional differences between mesenchymal stem cell populations are reflected by their transcriptome. *Stem cells and development*, *19*(4), 481–490. doi: 10.1089/scd.2009.0288

Kazman, R., Bianco, P., Ivers, J., & Klein, J. (2020, December). *Maintainability* (Report). Carnegie Mellon University. Retrieved 2024-03-07, from https://kilthub.cmu.edu/articles/report/Maintainability/12954908/1 doi: 10.1184/R1/12954908.v1

Kibler, C., Schermutzki, F., Waller, H. D., Timpl, R., Müller, C. A., & Klein, G. (1998, June). Adhesive interactions of human multiple myeloma cell lines with different extracellular matrix molecules. *Cell Adhesion and Communication*, *5*(4), 307–323. doi: 10.3109/15419069809040300

Krekel, H., Oliveira, B., Pfannschmidt, R., Bruynooghe, F., Laugher, B., & Bruhin, F. (2004). *Pytest.* Retrieved from https://github.com/pytest-dev/pytest

McKinney, W. (2010, January). Data Structures for Statistical Computing in Python. In (pp. 56–61). doi: 10.25080/Majora-92bf1922-00a

Moreno-Indias, I., Lahti, L., Nedyalkova, M., Elbere, I., Roshchupkin, G., Adilovic, M., … Zomer, A. L. (2021, February). Statistical and Machine Learning Techniques in Human Microbiome Studies: Contemporary Challenges and Solutions. *Frontiers in Microbiology*, *12*. Retrieved 2024-03-18, from https://www.frontiersin.org/journals/microbiology/articles/10.3389/fmicb.2021.635781/full doi: 10.3389/fmicb.2021.635781

Muruganandan, S., Roman, A. A., & Sinal, C. J. (2009, January). Adipocyte differentiation of bone marrow-derived mesenchymal stem cells: Cross talk with the osteoblastogenic program. *Cellular and molecular life sciences : CMLS*, *66*(2), 236–253. doi: 10.1007/s00018-008-8429-z

Myers, G. J., Sandler, C., & Badgett, T. (2011). *The art of software testing* (3rd ed.). Wiley Publishing. Retrieved from https://malenezi.github.io/malenezi/SE401/Books/114-the-art-of-software-testing-3-edition.pdf

Narzt, W., Pichler, J., Pirklbauer, K., & Zwinz, M. (1998, January). A Reusability Concept for Process Automation Software..

Nowotschin, S., & Hadjantonakis, A.-K. (2010, August). Cellular dynamics in the early mouse embryo: From axis formation to gastrulation. *Current opinion in genetics & development*, *20*(4), 420–427. doi: 10.1016/j.gde.2010.05.008

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., … Chintala, S. (2019, December). *PyTorch: An Imperative Style, High-Performance Deep Learning Library* (No. arXiv:1912.01703). arXiv. Retrieved 2024-03-07, from http://arxiv.org/abs/1912.01703 doi: 10.48550/arXiv.1912.01703

Peng, R. D. (2011, December). Reproducible Research in Computational Science. *Science*, *334*(6060), 1226–1227. Retrieved 2024-03-18, from https://www.science.org/doi/10.1126/science.1213847 doi: 10.1126/science.1213847

Perneger, T. V. (1998, April). What's wrong with Bonferroni adjustments. *BMJ : British Medical Journal*, *316*(7139), 1236–1238. Retrieved 2021-11-24, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1112991/

Pittenger, M. F., Mackay, A. M., Beck, S. C., Jaiswal, R. K., Douglas, R., Mosca, J. D., … Marshak, D. R.

(1999). Multilineage Potential of Adult Human Mesenchymal Stem Cells. , *284*(April), 143–148. doi: 10.1126/science.284.5411.143

The Python Language Reference. (n.d.). Retrieved 2024-03-07, from https://docs.python.org/3/reference/index.html

R Core Team. (2018). *R: A language and environment for statistical computing* [Manual]. Vienna, Austria. Retrieved from https://www.R-project.org/

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners.. Retrieved 2024-03-07, from https://www.semanticscholar.org/paper/Language-Models-are-Unsupervised-Multitask-Learners-Radford-Wu/9405cc0d6169988371b2755e573cc28650d14dfe

Rajkumar, S. V., & Kumar, S. (2020, September). Multiple myeloma current treatment algorithms. *Blood Cancer Journal*, *10*(9), 94. Retrieved 2023-07-03, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7523011/ doi: 10.1038/s41408-020-00359-2

Rayhan, A., & Gross, D. (2023). *The Rise of Python: A Survey of Recent Research.* doi: 10.13140/RG.2.2.27388.92809

Sacchetti, B., Funari, A., Remoli, C., Giannicola, G., Kogler, G., Liedtke, S., … Bianco, P. (2016). No identical "mesenchymal stem cells" at different times and sites: Human committed progenitors of distinct origin and differentiation potential are incorporated as adventitial cells in microvessels. *Stem Cell Reports*, *6*(6), 897–913. Retrieved from http://dx.doi.org/10.1016/j.stemcr.2016.05.011 doi: 10.1016/j.stemcr.2016.05.011

Sandve, G. K., Nekrutenko, A., Taylor, J., & Hovig, E. (2013, October). Ten Simple Rules for Reproducible Computational Research. *PLoS Computational Biology*, *9*(10), e1003285. Retrieved 2024-03-07, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3812051/ doi: 10.1371/journal.pcbi.1003285

Shenghui, H., Nakada, D., & Morrison, S. J. (2009). Mechanisms of Stem Cell Self-Renewal. *Annual Review of Cell and Developmental Biology*, *25*(1), 377–406. Retrieved from https://doi.org/10.1146/annurev.cellbio.042308.113248 doi: 10.1146/annurev.cellbio.042308.113248

Smith, A. M., Niemeyer, K. E., Katz, D. S., Barba, L. A., Githinji, G., Gymrek, M., … Vanderplas, J. T. (2018). Journal of Open Source Software (JOSS): Design and first-year review. *PeerJ Preprints*, *4*, e147. doi: 10.7717/peerj-cs.147

Stock, P., Bruckner, S., Winkler, S., Dollinger, M. M., & Christ, B. (2014, April). Human bone marrow mesenchymal stem cell-derived hepatocytes improve the mouse liver after acute acetaminophen intoxication by preventing progress of injury. *International journal of molecular sciences*, *15*(4), 7004–7028. doi: 10.3390/ijms15047004

Tam, P. P., & Beddington, R. S. (1987, January). The formation of mesodermal tissues in the mouse embryo during gastrulation and early organogenesis. *Development (Cambridge, England)*, *99*(1), 109–126.

Tanavalee, C., Luksanapruksa, P., & Singhatanadgige, W. (2016, June). Limitations of Using Microsoft Excel Version 2016 (MS Excel 2016) for Statistical Analysis for Medical Research. *Clinical Spine Surgery*, *29*(5), 203. Retrieved 2024-03-11, from https://journals.lww.com/jspinaldisorders/fulltext/2016/06000/limitations_of_using_microsoft_excel_version_2016.5.aspx doi: 10.1097/BSD.0000000000000382

Team, T. P. D. (2020, February). *Pandas-dev/pandas: Pandas.* Zenodo. Retrieved from https://doi.org/10.5281/zenodo.3509134 doi: 10.5281/zenodo.3509134

Terpos, E., Ntanasis-Stathopoulos, I., Gavriatopoulou, M., & Dimopoulos, M. A. (2018, January). Pathogenesis of bone disease in multiple myeloma: From bench to bedside. *Blood Cancer Journal*, *8*(1), 7. doi: 10.1038/s41408-017-0037-4

Ullah, I., Subbarao, R. B., & Rho, G. J. (2015). Human mesenchymal stem cells - current trends and future prospective Bioscience Reports.
doi: 10.1042/BSR20150025

Vallat, R. (2018, November). Pingouin: Statistics in Python. *Journal of Open Source Software*, *3*(31), 1026. Retrieved 2023-05-29, from https://joss.theoj.org/papers/10.21105/joss.01026 doi: 10.21105/joss.01026

van Rossum, G., Lehtosalo, J., & Langa, L. (2014). *PEP 484 – Type Hints | peps.python.org.* Retrieved 2024-03-08, from https://peps.python.org/pep-0484/

Viguet-Carrin, S., Garnero, P., & Delmas, P. D. (2006, March). The role of collagen in bone strength. *Osteoporosis International*, *17*(3), 319–336. Retrieved 2023-12-20, from https://doi.org/10.1007/s00198-005-2035-9 doi: 10.1007/s00198-005-2035-9

Waskom, M. L. (2021, April). Seaborn: Statistical data visualization. *Journal of Open Source Software*, *6*(60), 3021. Retrieved 2023-03-26, from https://joss.theoj.org/papers/10.21105/joss.03021 doi: 10.21105/joss.03021

Wickham, H. (2014, September). Tidy Data. *Journal of Statistical Software*, *59*, 1–23. Retrieved 2023-11-15, from https://doi.org/10.18637/jss.v059.i10 doi: 10.18637/jss.v059.i10

Wilkins, A., Kemp, K., Ginty, M., Hares, K., Mallam, E., & Scolding, N. (2009, July). Human bone marrow-derived mesenchymal stem cells secrete brain-derived neurotrophic factor which promotes neuronal survival in vitro. *Stem cell research*, *3*(1), 63–70. doi: 10.1016/j.scr.2009.02.006

Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., … Mons, B. (2016, March). The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, *3*(1), 160018. Retrieved 2024-03-18, from https://www.nature.com/articles/sdata201618 doi: 10.1038/sdata.2016.18

Xu, W., Zhang, X., Qian, H., Zhu, W., Sun, X., Hu, J., … Chen, Y. (2004, July). Mesenchymal stem cells from adult human bone marrow differentiate into a cardiomyocyte phenotype in vitro. *Experimental biology and medicine (Maywood, N.J.)*, *229*(7), 623–631.

Yang, A., Troup, M., & Ho, J. W. (2017, July). Scalability and Validation of Big Data Bioinformatics Software. *Computational and Structural Biotechnology Journal*, *15*, 379–386. Retrieved 2024-03-07, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5537105/ doi: 10.1016/j.csbj.2017.07.002

Appendices

A Supplementary Figures & Tables

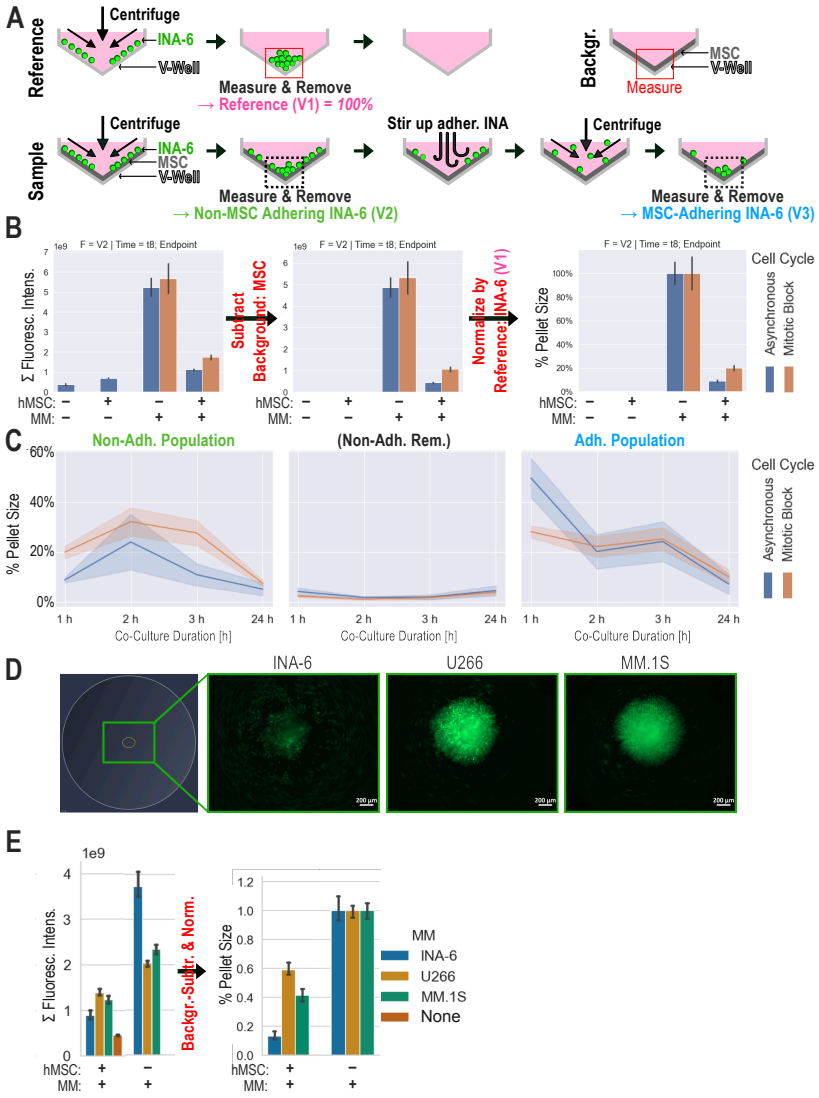


Figure 2: lorem ipsum

B Supplementary Materials and Methods

Lorem ipsum dolor

C Documentation of plotastic

C.1 Class Diagram

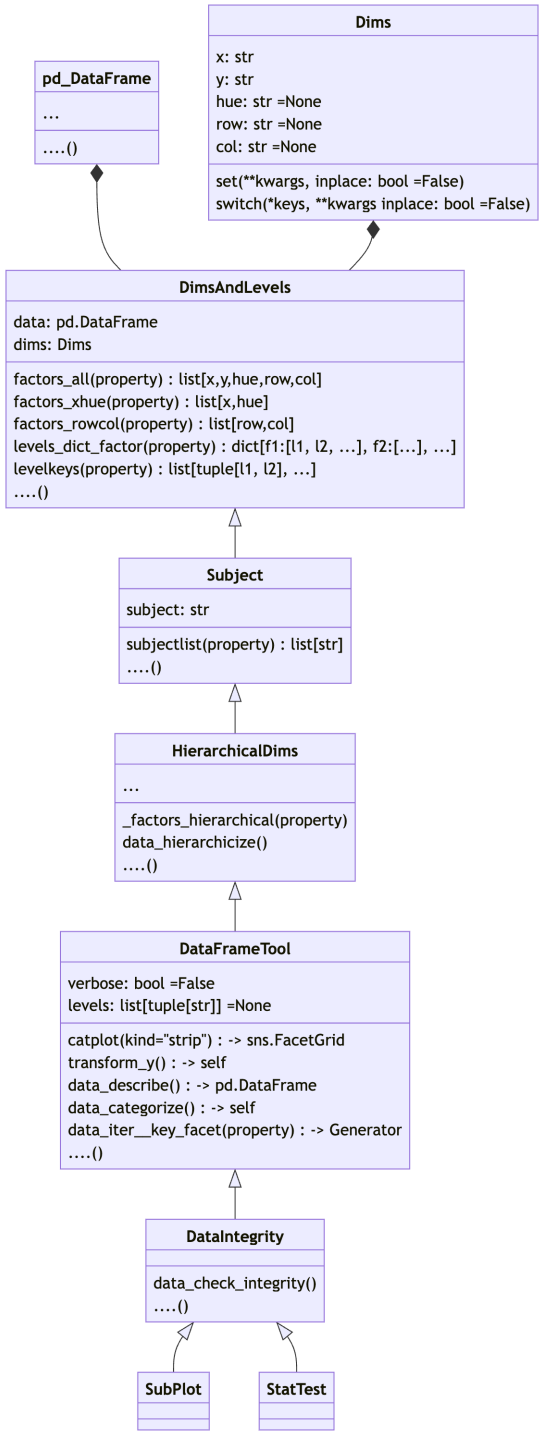


Figure 3: **Class diagram of plotastic (upper part):** The architecture of plotastic begins with classes that are related to handling a pandas.DataFrame object which stores the data, and defining dimensions to group the data (y, x, hue, col, row). This diagram ends with the classes SubPlot and StatTest and is continued on the next page. Arrow shapes follow the UML (unified modeling language): A hollow triangle indicates inheritance (*"is a"*) and a filled diamond indicates composition (*"has a"*).

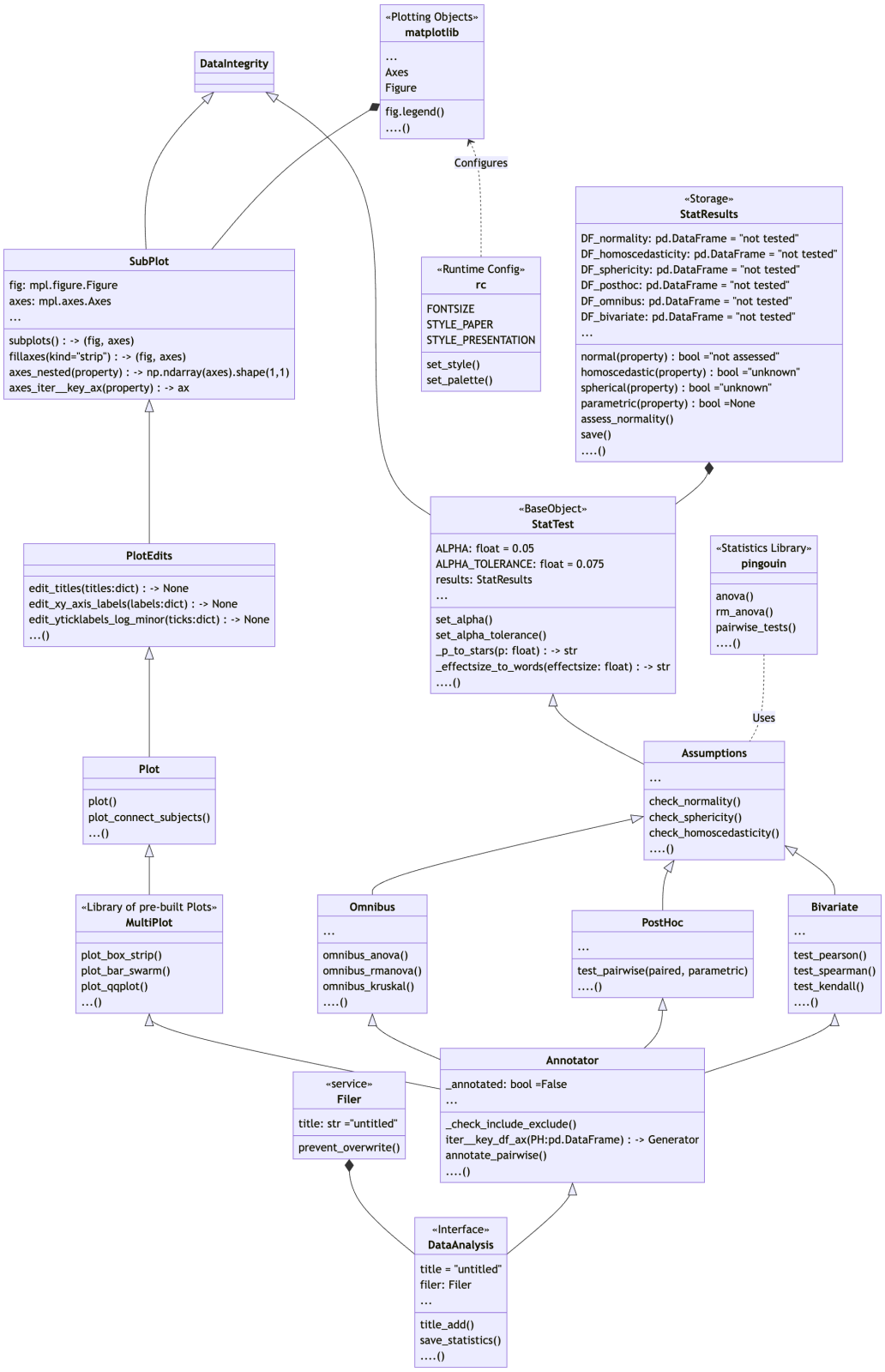
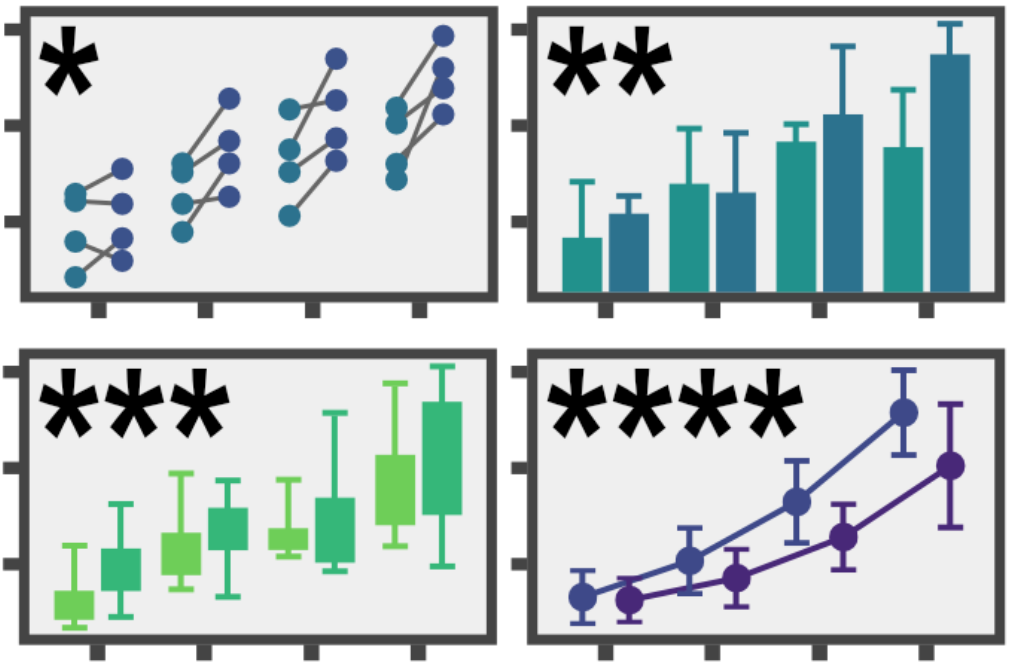


Figure 1: **(continued)** The architecture of `plotastic` continues after the class `DataIntegrity` with classes for plotting (`SubPlot`) and statistical testing (`StatTest`) and end with the class `DataAnalysis`, which serves as the main user interface. Arrow shapes follow the UML (unified modeling language): A hollow triangle indicates inheritance (*"is a"*) and a filled diamond indicates composition (*"has a"*).

C.2 Readme

The following pages are the README.md of plotastic found in the Python Package Index (PyPi) (pypi.org/project/plotastic), and on GitHub (github.com/markur4/plotastic).



---

plotastic: Bridging Plotting and Statistics

📦 Installation

Install from PyPi:

```
pip install plotastic
```

Install from GitHub: (experimental, check CHANGELOG.md)

```
pip install git+https://github.com/markur4/plotastic.git
```

Requirements

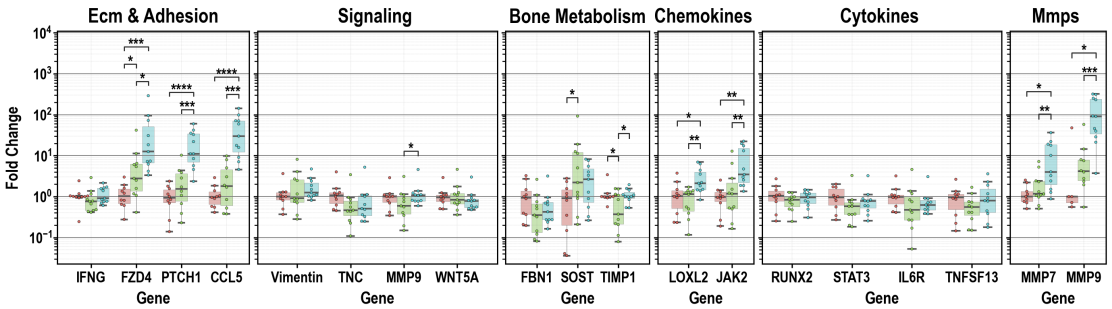
- Python >= 3.11 *(not tested with earlier versions)*

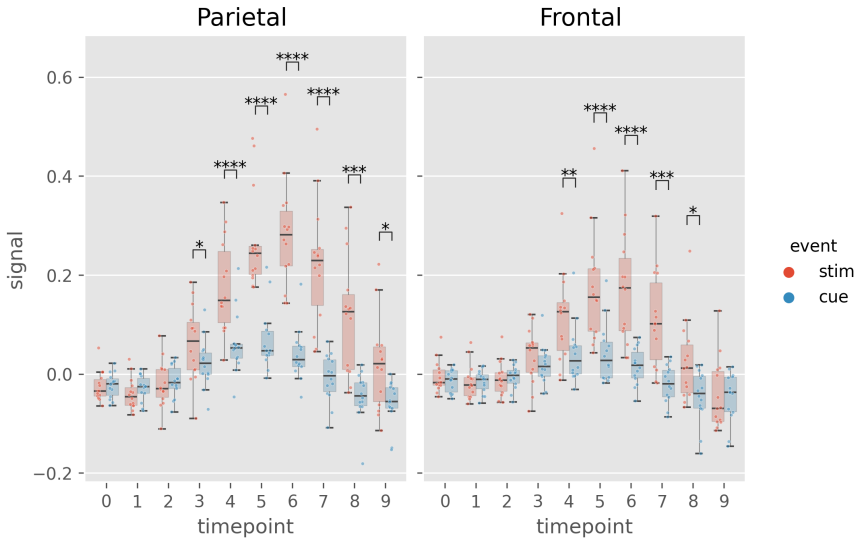
- pandas == 1.5.3 *(pingouin needs this)*
- seaborn <= 0.12.2 *(later versions reworked hue)*

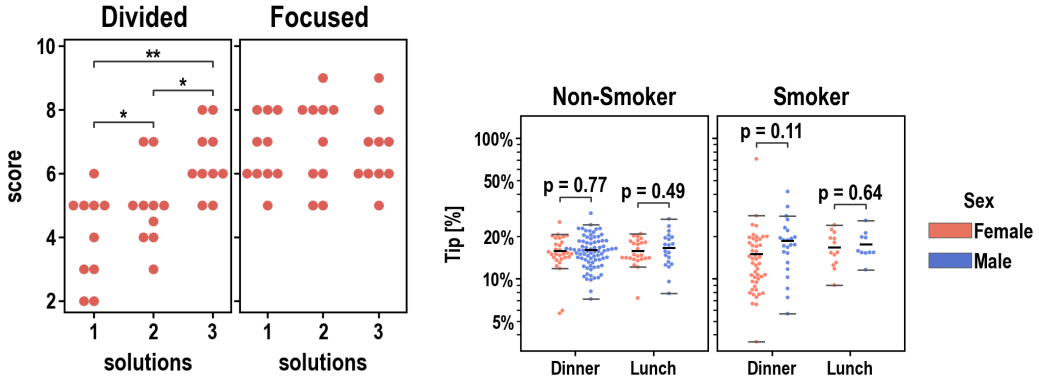
📷 Example Gallery

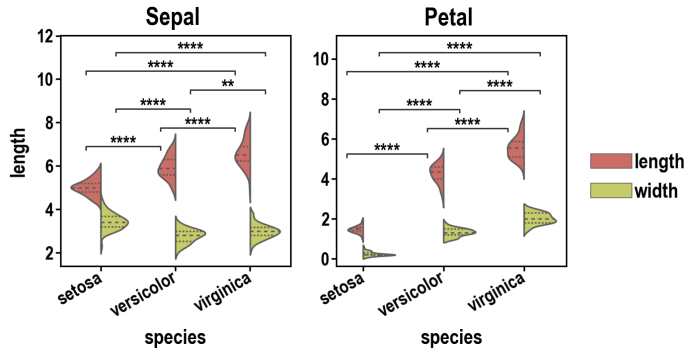
▶ ***(click to unfold)***

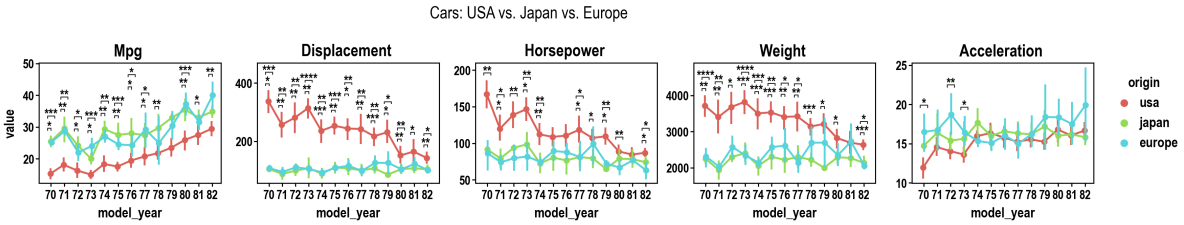
🐁 Click on Images for Code! 🐁

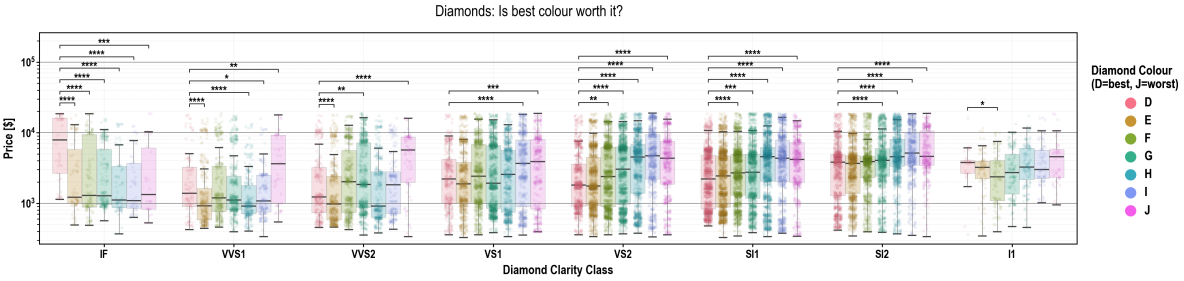


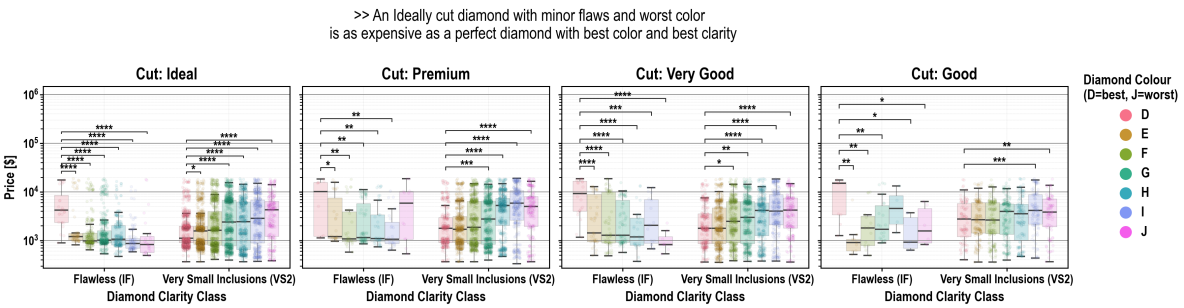












🤦 About plotastic

▶ 🤔 ***Summary***

plotastic addresses the challenges of transitioning from exploratory data analysis to hypothesis testing in Python's data science ecosystem. Bridging the gap between seaborn and pingouin, this library offers a unified environment for plotting and statistical analysis. It simplifies the workflow with a user-friendly syntax and seamless integration with familiar seaborn parameters (y, x, hue, row, col). Inspired by seaborn's consistency, plotastic utilizes a DataAnalysis object to intelligently pass parameters to pingouin statistical functions. The library systematically groups the data according to the needs of statistical tests and plots, conducts visualisation, analyses and supports extensive customization options. In essence, plotastic establishes a protocol for configuring statical analyses through plotting parameters.

This approach streamlines the process, translating `seaborn` parameters into statistical terms, providing researchers and data scientists with a cohesive and user-friendly solution in python.!

Workflow:

1. 🧱 **Import & Prepare your pandas DataFrame**
   - We require a long-format pandas dataframe with categorical columns
   - If it works with seaborn, it works with plotastic!
2. 🔀 **Make a DataAnalysis Object**
   - `DataAnalysis(DataFrame, dims={x, y, hue, row, col})`
   - Check for empty data groups, differing samplesizes, NaN-count, etc. automatically
3. ✅ **Explore Data**
   - Check Data integrity, unequal samplesizes, empty groups, etc.
   - Quick preliminary plotting with e.g. `DataAnalysis.catplot()`
4. 🔨 **Adapt Data**
   - Categorize multiple columns at once
   - Transform dependent variable
   - Each step warns you, if you introduced NaNs without knowledge!
   - etc.
5. ✨ **Perform Statistical Tests** ✨
   - Check Normality, Homoscedasticity, Sphericity
   - Perform Omnibus tests (ANOVA, RMANOVA, Kruskal-Wallis, Friedman)
   - Perform PostHoc tests (Tukey, Dunn, Wilcoxon, etc.) based on `pg.pairwise_tests()`
6. 📊 **Plot figure**
   - Use pre-defined and optimized multi-layered plots with one line (e.g. strip over box)!
   - Annotate statistical results (p-values as *, **, ***, etc.) with full control over which data to include or exclude!
7. 💿 **Save all results at once!**
   - One DataAnalysis object holds:
     - One DataFrame in `self.data`
     - One Figure in `self.fig`, `self.axes`
     - Multiple statistical results: `self.results`
   - Use `DataAnalysis.save_statistics()` to save all results to different sheets collected in one .xlsx filesheet per test

▶ 📊 ***Translating Plots into Statistics!***

In Principle:

- Categorical data is separable into `seaborn`'s categorization parameters: ***x***, ***y***, ***hue***, ***row***, ***col***. We call those *"dimensions"*.
- These dimensions are assigned to statistical terms:
  - ***y*** is the ***dependent variable*** (***DV***)
  - ***x*** and ***hue*** are ***independent variables*** (***IV***) and are treated as ***within/between factors*** (categorical variables)
  - ***row*** and ***col*** are ***grouping variables*** (categorical variables)
  - A ***subject*** may be specified for within/paired study designs (categorical variable)

- For each level of ***row*** or ***col*** (or for each combination of ***row***- and ***col*** levels), statistical tests will be performed with regards to the two-factors ***x*** and ***hue***

Example with ANOVA:

- Imagine this example data:
  - Each day you measure the tip of a group of people.
  - For each tip, you note down the ***day***, ***gender***, ***age-group*** and whether they ***smoke*** or not.
  - Hence, this data has 4 categorical dimensions, each with 2 or more *levels*:
    - ***day***: 4 levels (*monday*, *tuesday*, *wednesday*, *Thursday*)
    - ***gender***: 2 levels (*male*, *female*)
    - ***smoker***: 2 levels (*yes*, *no*)
    - ***age-group***: 2 levels (*young*, *old*)
- Each category is assigned to a place of a plot, and when calling statistical tests, we assign them to statistical terms (in comments):
  - ```
    # dims is short for dimensions
    dims = dict(          # STATISTICAL TERM:
        y = "tip",        # y-axis, dependent variable
        x = "day",        # x-axis, independent variable (within-
    subject factor)
        hue = "gender",   # color,  independent variable (within-
    subject factor)
        col = "smoker",   # axes,   grouping variable
        row = "age-group" # axes,   grouping variable
    )
    ```
- We perform statistical testing groupwise:
  - For each level-combinations of ***smoker*** and ***age-group***, a two-way ANOVA will be performed (with ***day*** and ***gender*** as ***between*** factors for each datagroup):
    - 1st ANOVA assesses datapoints where ***smoker***=*yes* AND ***age-group***=*young*
    - 2nd ANOVA assesses datapoints where ***smoker***=*yes* AND ***age-group***=*old*
    - 3rd ANOVA assesses datapoints where ***smoker***=*no* AND ***age-group***=*young*
    - 4th ANOVA assesses datapoints where ***smoker***=*no* AND ***age-group***=*old*
  - Three-way ANOVAs are not possible (yet), since that would require setting e.g. ***col*** as the third factor, or implementing another dimension (e.g. ***hue2***).

▶ ❗ ***Disclaimer about Statistics***

This software was inspired by ...

- ... ***Intuitive Biostatistics*** - Fourth Edition (2017); Harvey Motulsky
- ... ***Introduction to Statistical Learning with applications in Python*** - First Edition (2023); Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, Jonathan Taylor
- ... talking to other scientists struggling with statistics

✅ `plotastic` **can help you with...**
    ```

- ... gaining some practical experience when learning statistics
- ... quickly gain statistical implications about your data without switching to another software
- ... making first steps towards a full statistical analysis
- ... plotting publication grade figures (check statistics results with other software)
- ... publication grade statistical analysis **IF** you really know what you're doing OR you have back-checked your results by a professional statistician
- ... quickly test data transformations (log)

🚫 `plotastic` can NOT ...

- ... replace a professional statistician
- ... teach you statistics, you need some basic knowledge (but is awesome for practicing!)
- ... test for multicolinearity (Absence of multicolinearity is required by ANOVA!)
- ... perform stringent correction for multiple testing (e.g. bonferoni), as statistical tests are applied to sub-facets of the whole dataframe for each axes, which depends on the definition of x, hue, col, etc. Hence, corrected p-values might over-estimate the significance of your results.

🟡 Be critical and responsible with your statistical analysis!

- **Expect Errors:** Don't trust automated systems like this one!
- **Document your work in *ridiculous detail***:
  - Include the applied tests, the number of technical replicates and the number of biological/independent in each figure legend
  - State explicitly what each datapoint represents:
    - 1 datapoint = 1 Technical replicate?
    - 1 datapoint = The mean of all technical replicate per independent replicate/subject?
  - State explicitly what the error-bars mean: Standard deviation? Confidence interval?
  - (Don't mix technical with biological/independent variance)
  - Report if/how you removed outliers
  - Report if you did or did not apply correction methods (multiple comparisons, Greenhouse Geyser, etc.) and what your rationale is (exploratory vs. confirmatory study? Validation through other methods to reduce Type I error?)
- **Check results with professionnals:**
  - *"Here is my data, here is my question, here is my analysis, here is my interpretation. What do you think?"*

▶ ✅ *Feature List*

- ✅ **: Complete and tested**
- 👍 **: Complete**
- 📆 **: Planned or unfinished (no date)**
- 🤷‍♂️ **: Maybe..? (Rather not...)**
- 🚫 **: Not planned, don't want**
- 😖 **: Help Please..?**

▼ *Plotting*

- 👍 Make and Edit Plots: *Implemented* ✅

    - *All (non-facetgrid) seaborn plots should work, not tested*

- 📆 QQ-Plot
- 📆 Kaplan-Meyer-Plot
- 🤹 Interactive Plots (where you click stuff and adjust scale etc.)
    - *That's gonna be alot of work!*
- 🚫 Support for `seaborn.FacetGrid`
    - *Why not? - `plotastic` uses matplotlib figures and fills its axes with seaborn plot functions. In my opinion, that's the best solution that offers the best adaptibility of every plot detail while bieng easy to maintain*
- 🚫 Support for `seaborn.objects` (same as Facetgrid)
    - *Why not? - I don't see the need to refactor the code*
- 😒 **NEED HELP WITH:** The hidden state of `matplotlib` figures/plots/stuff that gets drawn:
    - *I want to save the figure in `DataAnalysis.fig` attribute. As simple as that sounds, matplotlib does weird stuff, not applying changes after editing the plot.*
    - *It'd be cool if I could control the changes to a DataAnalysis object better (e.g. using `inplace=True` like with `pd.DataFrames`). But I never figured out how to control matplotlib figure generation, even with re-drawing the figure with canvas. It's a mess and I wasted so much time already.*

▼ ***Multi-Layered Plotting***

- ✅ Box-plot + swarm
- 👍 Box-plot + strip
- 📆 Violin + swarm/strip

▼ ***Statistics***

- Assumption testing
    - ✅ Normality (e.g. Shapiro-Wilk)
    - ✅ Homoscedasticity (e.g. Levene)
    - ✅ Sphericity (e.g. Mauchly)
- Omnibus tests
    - ✅ ANOVA, RMANOVA, Kruskal-Wallis, Friedman
    - 📆 Mixed ANOVA
    - 📆 Annotate Results into Plot
- PostHoc
    - ✅ `pg.pairwise_tests()`
        - *Works with all primary options. That includes all parametric, non-parametric, paired, unpaired, etc. tests (t-test, paired t-test, MWU, Wilcoxon, etc.)*
    - ✅ Annotate Stars into plots (*, **, etc.)
        - *Specific pairs can be included/excluded from annotation*
    - 📆 Make correction for multiple testing go over complete DataFrame and not Facet-wise:
- Bivariate
    - 📆 Find and Implement system to switch between numerical and categorical x-axis
        - *Function to convert numerical data into categorical data by binning?*
    - 📆 Pearson, Spearman, Kendall

▼ ***Analysis Pipelines***

Idea: Put all those statistical tests into one line. I might work on this only after everything's implemented and working confidently and well!

- 🐝 `between_samples(parametric=True)`: ANOVA + Tukey (if Normality & Homoscedasticity are given)
- 🐝 `between_samples(parametric=False)`: Kruskal-Wallis + Dunn
- 🐝 `within_samples(parametric=True)`: RM-ANOVA + multiple paired t-tests (if Normality & Sphericity are given)
- 🐝 `within_samples(parametric=False)`: Friedman + multiple Wilcoxon

📔 How To Use

Documentations

1. **Example Gallery**
    1. Quick Example: FMRI
    2. qPCR (paired, parametric)
    3. Cars (unpaired, non-parametric)
    4. Diamonds (unpaired, non-parametric)
    5. Attention (paired/mixed, parametric)
    6. Iris (unpaired, parametric)
    7. Tips (unpaired, parametric)
2. **Data**
    1. Set/Switch Dimensions
3. **Plotting**
    1. Quick & Simple: MultiPlots
    2. Constructing Plots
    3. Legends
    4. Styles

Quick Example

Import plotastic and example Data

```
import matplotlib.pyplot as plt
import plotastic as plst

# Import Example Data (Long-Format)
DF, _dims = plst.load_dataset("fmri", verbose = False)
DF.head()
```

Assign each column to a dimension (y, x, hue, col, row):

```
dims = dict(
    y = "signal",    # y-axis, dependent variable
    x = "timepoint", # x-axis, independent variable & within-subject
factor
    hue = "event",   # color,  grouping variable & within-subject factor
    col = "region"   # axes,   grouping variable
)
```

Initialize DataAnalysis Object

```
DA = plst.DataAnalysis(
    data=DF,           # Dataframe, long format
    dims=dims,         # Dictionary with y, x, hue, col, row
    subject="subject", # Datapoints are paired by subject (optional)
    verbose=False,     # Print out info about the Data (optional)
)
```

Perform Statistics

No arguments need to be passed, although `**kwargs`, are passed to respective `pingouin` functions.

```
DA.check_normality()  # Normal Distribution?
DA.check_sphericity() # Sphericity?
DA.omnibus_rm_anova() # Repeated Measures ANOVA
DA.test_pairwise()    # Post-hoc tests
```

Save Results:

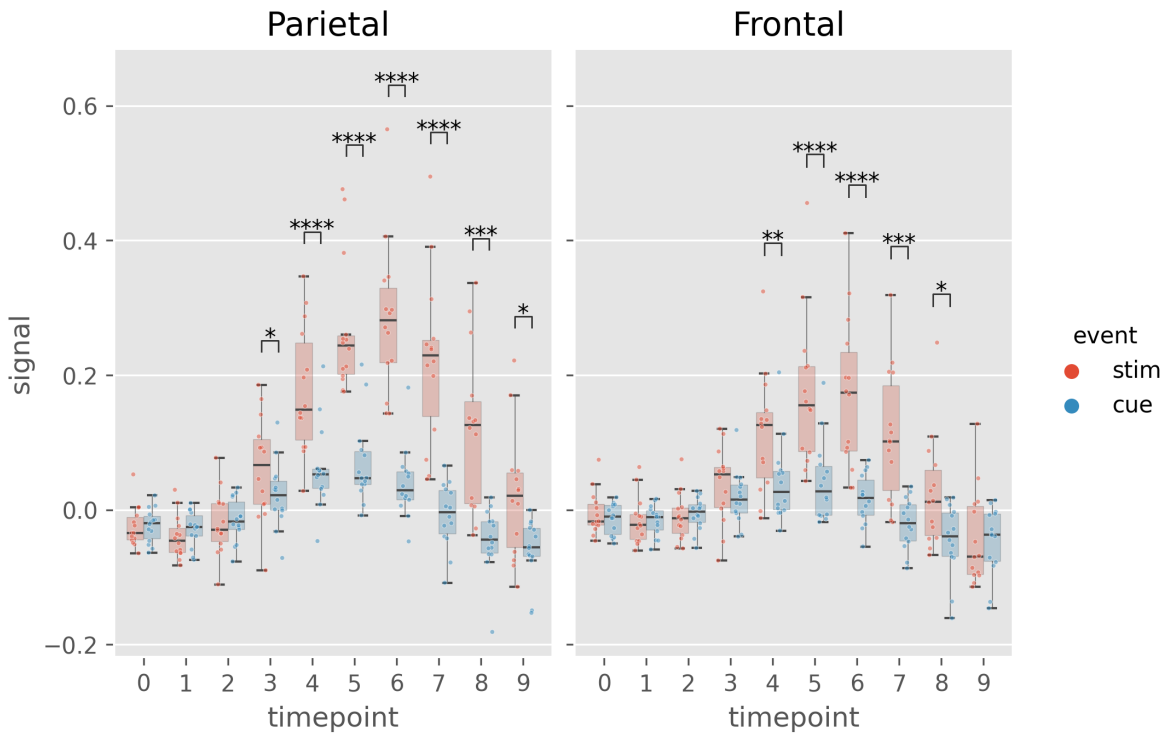
Output is one excel file containing results of all performed tests (normality, anova, t-tests, etc.) in different sheets

```
DA.save_statistics("example.xlsx")
```

Annotate post-hoc results into plot:

```
(DA
 .plot_box_strip()   # Pre-built plotting function initializes plot
 .annotate_pairwise( # Annotate results from DA.test_pairwise()
     include="__HUE" # Use only significant pairs across each hue
     )
)
```

```
# Saving the plot like matplotlib!
plt.savefig("example.png", dpi=200, bbox_inches="tight")
```



🧪 Testing

▶ ***(click to unfold)***

- Download/Clone repository
- Install development tools `pip install .[dev]`
- Run tests
  - Run `pytest ./tests`
  - To include a coverage report run `pytest ./tests --cov--cov-report=html` and open `./htmlcov/index.html` with your browser.

🤝 Community Guidelines

▶ ***(click to unfold)***

When interacting with the community, you must adhere to the Code of Conduct

Contribute

I am grateful for pull requests!

- Make sure to understand the code (e.g. see Class diagram in this Readme)
- Run tests before submitting a pull request

Reporting Issues & Problems

If you need help, please open an issue on this repository.

- Please provide a minimal example to reproduce the problem.

Support

If you need help, please open an issue on this repository.

✍🏻 Cite These!

▶ ***(click to unfold)***

Kuric et al., (2024). plotastic: Bridging Plotting and Statistics in Python. Journal of Open Source Software, 9(95), 6304, https://doi.org/10.21105/joss.06304

Vallat, R. (2018). Pingouin: statistics in Python. Journal of Open Source Software, 3(31), 1026, https://doi.org/10.21105/joss.01026

Waskom, M. L., (2021). seaborn: statistical data visualization. Journal of Open Source Software, 6(60), 3021, https://doi.org/10.21105/joss.03021.

```
@article{Kuric2024,
  doi = {10.21105/joss.06304},
  url = {https://doi.org/10.21105/joss.06304},
  year = {2024}, publisher = {The Open Journal},
  volume = {9},
  number = {95},
  pages = {6304},
  author = {Martin Kuric and Regina Ebert},
  title = {plotastic: Bridging Plotting and Statistics in Python},
  journal = {Journal of Open Source Software}
}


@article{Waskom2021,
    doi = {10.21105/joss.03021},
    url = {https://doi.org/10.21105/joss.03021},
    year = {2021},
    publisher = {The Open Journal},
    volume = {6},
    number = {60},
    pages = {3021},
    author = {Michael L. Waskom},
    title = {seaborn: statistical data visualization},
    journal = {Journal of Open Source Software}
 }

 @article{Vallat2018,
  title    = "Pingouin: statistics in Python",
  author   = "Vallat, Raphael",
  journal  = "The Journal of Open Source Software",
  volume   =  3,
```

```
    number   =  31,
    pages    = "1026",
    month    =  nov,
    year     =  2018
  }
```

C.3 Example Analysis “qpcr”

The following pages are a jupyter notebook from an example analysis using `plotastic` that's found on GitHub (`github.com/markur4/plotastic`). The Dataset is derived from Chapter 1 qPCR of this thesis, exchanging the original gene names with random ones, while preserving gene classes.

qpcr

April 9, 2024

[ ]:
```
import plotastic as plst
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

[ ]:
```
# Set Plot Style
plst.set_style("paper")
# plst.set_palette("hls", verbose=True)
plst.set_palette(["#db5f57", "#91db57", "#57d3db"])
```

```
#! You chose this color palette: ['#db5f57', '#91db57', '#57d3db', '#db5f57',
'#91db57', '#57d3db', '#db5f57', '#91db57']
```

```
['#db5f57',
 '#91db57',
 '#57d3db',
 '#db5f57',
 '#91db57',
 '#57d3db',
 '#db5f57',
 '#91db57']
```

1 Example Analysis: qPCR

Raw Data: https://github.com/markur4/plotastic/tree/main/src/plotastic/example_data/data

Original Source: (unpublished)

[ ]:
```
# Import Example Data
DF, _dims = plst.load_dataset("qpcr", verbose=False)
dims = dict(
    y="fc",
    x="gene",
    hue="fraction",
    # col= 'method',
    row="class",
)
DA = plst.DataAnalysis(DF, dims, subject="subject", verbose=False)
```

[ ]:
```
DA.transform_y("log10", inplace=True)  # Log transform
DA.check_normality()  # -> Only few groups are not normal -> parametric
```

[ ]:

class	gene	fraction	W	pval	normal	n
Bone Metabolism	F1	FBN1	0.936873	0.518768	True	10
		SOST	0.880395	0.131862	True	10
		TIMP1	0.745494	0.004807	False	9
	F2	FBN1	0.954764	0.705148	True	11
		SOST	0.967810	0.863610	True	11
		TIMP1	0.914325	0.274168	True	11
	F3	FBN1	0.915247	0.281020	True	11
		SOST	0.923112	0.345415	True	11
		TIMP1	0.937230	0.488505	True	11
Chemokines	F1	LOXL2	0.930358	0.451421	True	10
		JAK2	0.897331	0.204749	True	10
	F2	LOXL2	0.874630	0.088876	True	11
		JAK2	0.960025	0.772006	True	11
	F3	LOXL2	0.943678	0.564652	True	11
		JAK2	0.878406	0.099301	True	11
Cytokines	F1	RUNX2	0.947142	0.634825	True	10
		STAT3	0.933422	0.482382	True	10
		IL6R	0.927258	0.421472	True	10
		TNFSF13	0.907481	0.264130	True	10
	F2	RUNX2	0.915611	0.283765	True	11
		STAT3	0.907354	0.226836	True	11
		IL6R	0.985709	0.989621	True	11
		TNFSF13	0.958855	0.757330	True	11
	F3	RUNX2	0.924060	0.353917	True	11
		STAT3	0.932663	0.438418	True	11
		IL6R	0.826181	0.020798	False	11
		TNFSF13	0.970421	0.890746	True	11
ECM & Adhesion	F1	IFNG	0.715267	0.001349	False	10
		FZD4	0.981633	0.973303	True	10
		PTCH1	0.911578	0.292008	True	10
		CCL5	0.969121	0.882582	True	10
	F2	IFNG	0.899109	0.180269	True	11
		FZD4	0.979590	0.963841	True	11
		PTCH1	0.986610	0.990734	True	10
		CCL5	0.925780	0.407685	True	10
	F3	IFNG	0.905665	0.216509	True	11
		FZD4	0.923819	0.351743	True	11
		PTCH1	0.957827	0.744318	True	11
		CCL5	0.940093	0.521596	True	11
MMPs	F1	MMP7	0.955749	0.752957	True	9
		MMP9	0.675286	0.005186	False	5
	F2	MMP7	0.926078	0.372552	True	11

```
                     MMP9      0.971128  0.901100    True  10
               F3    MMP7      0.924886  0.361455    True  11
                     MMP9      0.913554  0.268549    True  11
  Signaling        F1 Vimentin  0.919696  0.354424    True  10
                     TNC       0.928589  0.434161    True  10
                     NOTCH1    0.922084  0.374662    True  10
                     WNT5A     0.903581  0.239742    True  10
               F2    Vimentin  0.957763  0.743507    True  11
                     TNC       0.959813  0.769352    True  11
                     NOTCH1    0.977556  0.951045    True  11
                     WNT5A     0.937156  0.487661    True  11
               F3    Vimentin  0.910924  0.250109    True  11
                     TNC       0.884194  0.117578    True  11
                     NOTCH1    0.779982  0.005132   False  11
                     WNT5A     0.812114  0.013581   False  11
```

[ ]:
```
DA.check_sphericity()
```

[ ]:
```
                                spher         W       chi2  dof      pval  \
class           fraction
Bone Metabolism F1       0   True  0.592922   3.658847    2  0.160506
                F2       0   True  0.703252   3.168356    2  0.205116
                F3       0   True  0.832864   1.645964    2  0.439120
Chemokines      F1       0   True       NaN        NaN    1  1.000000
                F2       0   True       NaN        NaN    1  1.000000
                F3       0   True       NaN        NaN    1  1.000000
Cytokines       F1       0   True  0.629185   3.577934    5  0.614197
                F2       0  False  0.262747  11.657816    5  0.040987
                F3       0  False  0.210032  13.610980    5  0.019012
ECM & Adhesion  F1       0   True  0.486690   5.560987    5  0.354712
                F2       0   True  0.295164   8.202615    5  0.149255
                F3       0   True  0.297080  10.586623    5  0.061736
MMPs            F1       0   True       NaN        NaN    1  1.000000
                F2       0   True       NaN        NaN    1  1.000000
                F3       0   True       NaN        NaN    1  1.000000
Signaling       F1       0   True  0.536227   4.812474    5  0.442437
                F2       0   True  0.554009   5.151113    5  0.400336
                F3       0  False  0.117602  18.669462    5  0.002375

                                group count       n per group
class           fraction
Bone Metabolism F1       0            3       [10, 10, 9]
                F2       0            3      [11, 11, 11]
                F3       0            3      [11, 11, 11]
Chemokines      F1       0            2          [10, 10]
                F2       0            2          [11, 11]
                F3       0            2          [11, 11]
```

```
Cytokines       F1       0            4  [10, 10, 10, 10]
                F2       0            4  [11, 11, 11, 11]
                F3       0            4  [11, 11, 11, 11]
ECM & Adhesion  F1       0            4  [10, 10, 10, 10]
                F2       0            4  [10, 11, 11, 10]
                F3       0            4  [11, 11, 11, 11]
MMPs            F1       0            2           [9, 5]
                F2       0            2         [11, 10]
                F3       0            2         [11, 11]
Signaling       F1       0            4  [10, 10, 10, 10]
                F2       0            4  [11, 11, 11, 11]
                F3       0            4  [11, 11, 11, 11]
```

[ ]:
```
# Default is (paired) t-test, and since DA has subject: paired=True
DA.test_pairwise()
```

[ ]:
```
                                          gene     A      B   mean(A)  \
class          fraction Contrast
ECM & Adhesion -        gene                 -  CCL5   FZD4  0.591713
                        gene                 -  CCL5   IFNG  0.591713
                        gene                 -  CCL5  PTCH1  0.591713
                        gene                 -  FZD4   IFNG  0.622994
                        gene                 -  FZD4  PTCH1  0.622994
…                                          …  …    …       …
MMPs           NaN      gene * fraction   MMP9    F1     F3  0.256111
                        gene * fraction   MMP9    F2     F3  0.677357
               F1       fraction * gene    NaN  MMP7   MMP9  0.032549
               F2       fraction * gene    NaN  MMP7   MMP9  0.185211
               F3       fraction * gene    NaN  MMP7   MMP9  0.742060

                                           std(A)   mean(B)    std(B) Paired  \
class          fraction Contrast
ECM & Adhesion -        gene             0.253752  0.622994  0.266747   True
                        gene             0.253752 -0.026656  0.149430   True
                        gene             0.253752  0.469495  0.330886   True
                        gene             0.266747 -0.026656  0.149430   True
                        gene             0.266747  0.469495  0.330886   True
…                                           …        …        …    …
MMPs           NaN      gene * fraction  0.802159  1.845550  0.600687   True
                        gene * fraction  0.546148  1.845550  0.600687   True
               F1       fraction * gene  0.228544  0.256111  0.802159   True
               F2       fraction * gene  0.361750  0.677357  0.546148   True
               F3       fraction * gene  0.567249  1.845550  0.600687   True

                                          Parametric          T   dof  \
class          fraction Contrast
ECM & Adhesion -        gene                 True  -0.327586  10.0
```

```
                        gene                  True   7.882620  10.0
                        gene                  True   1.320783  10.0
                        gene                  True   7.532512  10.0
                        gene                  True   2.105924  10.0
…                                            …        …   …
MMPs            NaN     gene * fraction       True  -3.513968   4.0
                        gene * fraction       True  -5.680475   9.0
                F1      fraction * gene       True  -0.543884   4.0
                F2      fraction * gene       True  -3.811156   9.0
                F3      fraction * gene       True -15.767066  10.0

                                          alternative         p-unc       BF10  \
class           fraction Contrast
ECM & Adhesion -         gene               two-sided  7.499799e-01      0.312
                         gene               two-sided  1.339935e-05   1643.947
                         gene               two-sided  2.160003e-01      0.598
                         gene               two-sided  1.987311e-05   1165.781
                         gene               two-sided  6.146203e-02      1.461
…                                              …           …        …
MMPs            NaN      gene * fraction    two-sided  2.458360e-02      3.686
                         gene * fraction    two-sided  3.016844e-04    111.751
                F1       fraction * gene    two-sided  6.154168e-01      0.448
                F2       fraction * gene    two-sided  4.145762e-03     12.636
                F3       fraction * gene    two-sided  2.163081e-08  4.845e+05

                                            hedges **p-unc    Sign.  \
class           fraction Contrast
ECM & Adhesion -         gene             -0.115597      ns    False
                         gene              2.856874    ****  signif.
                         gene              0.398765      ns    False
                         gene              2.890772    ****  signif.
                         gene              0.491362   0.061   toler.
…                                             …     …      …
MMPs            NaN      gene * fraction -2.179426       *  signif.
                         gene * fraction -2.024511     ***  signif.
                F1       fraction * gene -0.255634      ns    False
                F2       fraction * gene -0.939861      **  signif.
                F3       fraction * gene -1.817138    ****  signif.

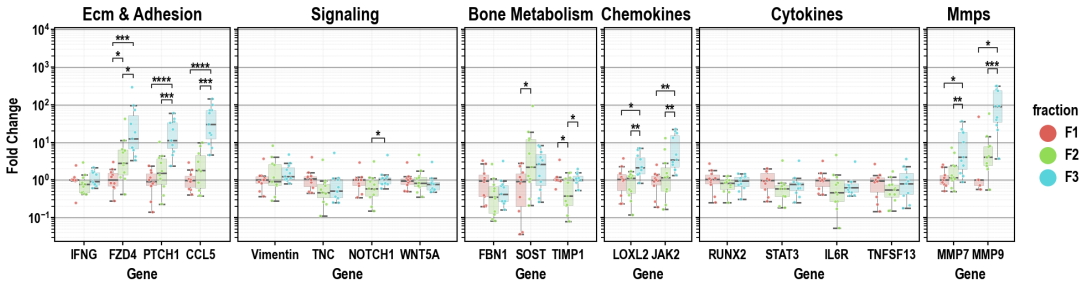
                                                          pairs cross
class           fraction Contrast
ECM & Adhesion -         gene                        (CCL5, FZD4)     x
                         gene                        (CCL5, IFNG)     x
                         gene                       (CCL5, PTCH1)     x
                         gene                        (FZD4, IFNG)     x
                         gene                       (FZD4, PTCH1)     x
…                                                          …   …
```

```
MMPs          NaN      gene * fraction  ((MMP9, F3), (MMP9, F1))   hue
                       gene * fraction  ((MMP9, F3), (MMP9, F2))   hue
              F1       fraction * gene  ((MMP9, F1), (MMP7, F1))     x
              F2       fraction * gene  ((MMP9, F2), (MMP7, F2))     x
              F3       fraction * gene  ((MMP9, F3), (MMP7, F3))     x

[167 rows x 19 columns]
```

[ ]:
```python
# Plot
(
    DA.switch("row", "col", verbose=False)
    .set(y="fc", inplace=False)  # set y back to to fc to display non-log values
    .plot_box_strip(
        subplot_kws=dict(
            figsize=(10, 2.5),
            width_ratios=[4, 5, 3, 2, 5, 2],
        ),
        strip_kws=dict(alpha=0.8),
    )
    .edit_grid()
    .edit_y_scale_log(10)
    .edit_xy_axis_labels(y_leftmost_col="Fold Change", x="Gene")
    .annotate_pairwise(include="__HUE")
)

plt.savefig("qpcr1.png", dpi=300, bbox_inches="tight")
```



D Submission Forms & Documents

D.1 Author Contributions

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Graduate School
Life Sciences

Statement of individual author contributions and of legal second publication rights to manuscripts included in the dissertation

Manuskript 1: Research Article (submitted, under revision)					
Martin Kuric (MK), Susanne Beck, Doris Schneider, Wyonna Rindt, Marietheres Evers, Jutta Meißner-Weigl, Sabine Zeck, Melanie Krug, Marietta Herrmann, Tanja Nicole Hartmann, Ellen Leich, Maximilian Rudert, Denitsa Docheva, Anja Seckinger, Dirk Hose, Franziska Jundt, Regina Ebert (RE) (2024): Keep it Together: Describing Myeloma Dissemination *in vitro* with hMSC-Interacting Subpopulations and their Aggregation/Detachment Dynamics, **Cancer Research Communications**					
Participated in	**Author Initials,** Responsibility decreasing from left to right				
Study Design	MK	Regina Ebert	Wyonna Rindt		
Methods Development	MK	Doris Schneider			
Data Collection	MK	Doris Schneider			
Data Analysis and Interpretation	MK	Susanne Beck	Regina Ebert		
Manuscript Writing Writing of Introduction Writing of Materials & Methods Writing of Discussion Writing of First Draft	MK	Regina Ebert			

Explanations: The content of this publication exceeds the usual scope (~29 pages Supplemental). It includes not only research findings but also survival data and protocols of new, established methods and their validations. The contribution of Martin Kuric was pivotal and predominant in all aspects of this work. Doris Schneider assisted in the experimental procedures. Susanne Beck analyzed the raw data from RNAseq and survival data, which were interpreted, depicted, and summarized by Martin Kuric.

Manuscript 2: Data Analysis Software (submitted, passed peer-review, under revision)					
Martin Kuric (MK), Regina Ebert (2024): plotastic: Bridging Plotting and Statistics in Python, **Journal of Open Source Software**					
Participated in	**Author Initials,** Responsibility decreasing from left to right				
Idea, Architectural Design	MK				
Software Development Feature Implementation Testing	MK				
Distribution of Software Documentation Version Control (GitHub) Deployment (PyPi)	MK				
Manuscript Writing Writing of Statement of Need Writing of Example Writing of Overview	MK	Regina Ebert			

Explanations: The software was entirely created by Martin Kuric, comprising more than 8000 total lines (including ~2000 testable lines) and is comparable in size to a typical web application. The release of this software involved version control using GitHub, packaging and deployment on PyPi. Regina Ebert gave feedback on submitted manuscript.

Manuscript 3: Research Letter (published)

Daniela Simone Maichl, Julius Arthur Kirner, Susanne Beck, Wen-Hui Cheng, Melanie Krug, Martin Kuric (MK), Carsten Patrick Ade, Thorsten Bischler, Franz Jakob, Dirk Hose, Anja Seckinger, Regina Ebert & Franziska Jundt (2023): Identification of NOTCH-driven matrisome-associated genes as prognostic indicators of multiple myeloma patient survival, **Blood Cancer Journal 13:134**

Participated in	**Author Initials,** Responsibility decreasing from left to right				
Study Design Methods Development	Daniela Simone		Franziska Jundt		
Data Collection	Daniela Simone		Franziska Jundt		
Data Analysis and Interpretation	Daniela Simone	Susanne Beck	Franziska Jundt	MK	
Manuscript Writing Writing of Introduction Writing of Materials & Methods Writing of Discussion Writing of First Draft	Daniela Simone		Franziska Jundt	MK	

Explanations: This co-authorship is not a chapter in this dissertation. Martin Kuric produced figures of processed but complex-to-visualize data and gave feedback on submitted manuscript.

Manuscript 4: Research Paper (under peer-review)

Wyonna Rindt, Melanie Krug, Shuntaro Yamada, Franziska Sennefelder, Louisa Belz, Wen-Hui Cheng, Azeem Muhammad, Martin Kuric (MK), Marietheres Evers, Ellen Leich, Tanja Nicole Hartmann, Ana Rita Pereira, Marietta Hermann, Jan Hansmann, Mohammed Ahmed Yassin, Kamal Mustafa, Regina Ebert, and Franziska Jundt (2024): A 3D bioreactor model to study osteocyte differentiation and mechanobiology under perfusion and compressive mechanical loading, **Acta Biomaterialia**

Participated in	**Author Initials,** Responsibility decreasing from left to right				
Study Design Methods Development	Wyonna Rindt	Franziska Jundt		MK	
Data Collection	Wyonna Rindt	Franziska Jundt		MK	
Data Analysis and Interpretation	Wyonna Rindt	Franziska Jundt		MK	
Manuscript Writing Writing of Introduction Writing of Materials & Methods Writing of Discussion Writing of First Draft	Wyonna Rindt	Franziska Jundt		MK	

Explanations: This co-authorship is not a chapter in this dissertation. Martin Kuric contributed by counseling during weekly meetings in tight collaboration with Franziska Jundt's group, assisting Wyonna Rindt during laboratory experiments, image analysis and giving feedback on submitted manuscript.

Manuscript 5: Research Paper (under revision)

Marietta Herrmann, Jutta Schneidereit, Susanne Wiesner, Martin Kuric (MK), Maximilian Rudert, Martin Lüdemann, Mugdha Srivastava, Norbert Schütze, Regina Ebert, Denitsa Docheva, Franz Jakob (2024): Peripheral blood cells enriched by adhesion to CYR61 are heterogenous myeloid modulators of tissue regeneration with early endothelial progenitor characteristics, **European Cells and Materials**

Participated in	**Author Initials,** Responsibility decreasing from left to right				
Study Design Methods Development	Marietta Herrmann				
Data Collection	Marietta Herrmann			MK	

Data Analysis and Interpretation	Marietta Herrmann				
Manuscript Writing Writing of Introduction Writing of Materials & Methods Writing of Discussion Writing of First Draft	Marietta Herrmann			MK	

Explanations: This co-authorship is not a chapter in this dissertation. Martin Kuric contributed by establishing and measuring large automated microscopy scans of stained cells for quantifying osteogenic differentiation and giving feedback on submitted manuscript.

Manuscript 6: Research Letter (published)

Marietheres Evers, Martin Schreder, Thorsten Stühmer, Franziska Jundt, Regina Ebert, Tanja Nicole Hartmann, Michael Altenbuchinger, Martina Rudelius, Martin Kuric (MK), Wyonna Darleen Rindt, Torsten Steinbrunn, Christian Langer, Sofia Catalina Heredia-Guerrero, Hermann Einsele, Ralf Christian Bargou, Andreas Rosenwald, Ellen Leich (2023): Prognostic value of extracellular matrix gene mutations and expression in multiple myeloma, **Blood Cancer J. 13(1):43**

Participated in	**Author Initials,** Responsibility decreasing from left to right				
Study Design Methods Development	Marietheres Evers				
Data Collection	Marietheres Evers				
Data Analysis and Interpretation	Marietheres Evers			MK	
Manuscript Writing Writing of Introduction Writing of Materials & Methods Writing of Discussion Writing of First Draft	Marietheres Evers			MK	

Explanations: This co-authorship is not a chapter in this dissertation. Martin Kuric contributed by counseling during regular meetings with Ellen Leich's group and giving feedback on submitted manuscript.

If applicable, the doctoral researcher confirms that she/he has obtained permission from both the publishers (copyright) and the co-authors for legal second publication.

The doctoral researcher and the primary supervisor confirm the correctness of the above mentioned assessment.

		Würzburg	
Doctoral Researcher's Name	Date	Place	Signature

		Würzburg	
Primary Supervisor's Name	Date	Place	Signature

D.2 Author Contributions to Figures and Tables

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Graduate School
Life Sciences

Statement of individual author contributions to figures/tables of manuscripts included in the dissertation

Manuskript 1: Research Article (submitted, under revision)

Martin Kuric (MK), Susanne Beck, Doris Schneider, Wyonna Rindt, Marietheres Evers, Jutta Meißner-Weigl, Sabine Zeck, Melanie Krug, Marietta Herrmann, Tanja Nicole Hartmann, Ellen Leich, Maximilian Rudert, Denitsa Docheva, Anja Seckinger, Dirk Hose, Franziska Jundt, Regina Ebert1 (2024): Keep it Together: Describing Myeloma Dissemination *in vitro* with hMSC-Interacting Subpopulations and their Aggregation/Detachment Dynamics, **Cancer Research Communications**

Figure	**Author Initials,** Responsibility decreasing from left to right				
1	MK	Doris Schneider			
2	MK	Doris Schneider			
3	MK	Doris Schneider	Sabine Zeck	Wyonna Rindt	Melanie Krug
4	MK	Doris Schneider	Susanne Beck		
5	MK	Susanne Beck			
6	MK	Susanne Beck			
7	MK				
S1	MK	Doris Schneider	Sabine Zeck	Wyonna Rindt	Melanie Krug
S2	MK	Doris Schneider	Marietta Herrmann		
S3	MK	Doris Schneider	Sabine Zeck		
S4	MK				
S5	MK				
S6	MK	Susanne Beck			
Table	**Author Initials,** Responsibility decreasing from left to right				
1	MK	Susanne Beck			
2	MK	Susanne Beck			
S1	MK	Doris Schneider			
S2	MK	Susanne Beck			
S3	MK				
S4	MK	Doris Schneider			

Manuscript 2: Data Analysis Software (submitted, passed peer-review, under revision)

Martin Kuric, Regina Ebert (2024): plotastic: Bridging Plotting and Statistics in Python, **Journal of Open Source Software**

Figure	**Author Initials,** Responsibility decreasing from left to right				
1	MK				
Table	**Author Initials,** Responsibility decreasing from left to right				
1	MK				
2	MK				

Documentation	**Author Initials,** Responsibility decreasing from left to right				
README	MK				
Example Gallery	MK				
Features	MK				
Testing	**Author Initials,** Responsibility decreasing from left to right				
Test-Code (Pytest)	MK				
Continuous Integration	MK				

Explanations: All files are available on GitHub (https://github.com/markur4/plotastic) and installable via pypi.com. Documentations are found in the Readme, including example gallery and feature explanation. Software tests was written using pytest. Coverage of code by tests is reviewable with codecov (https://app.codecov.io/gh/markur4/plotastic). Continuous Integration is implemented using GitHub actions.

Manuscript 3: Research Letter (published)

Daniela Simone Maichl, Julius Arthur Kirner, Susanne Beck, Wen-Hui Cheng, Melanie Krug, Martin Kuric, Carsten Patrick Ade, Thorsten Bischler, Franz Jakob, Dirk Hose, Anja Seckinger, Regina Ebert & Franziska Jundt (2023): Identification of NOTCH-driven matrisome-associated genes as prognostic indicators of multiple myeloma patient survival, **Blood Cancer Journal 13:134**

Figure	**Author Initials,** Responsibility decreasing from left to right				
1 a	Daniela Simone			Susanne Beck	
1 b	Daniela Simone			Susanne Beck	
1 c	Daniela Simone			Susanne Beck	MK
1 d	Daniela Simone			Susanne Beck	MK
Table	**Author Initials,** Responsibility decreasing from left to right				
1	Daniela Simone				

Explanations: Martin Kuric plotted multidimensional diagrams using python and fine-adjusted them using professional design software (Affinity Publisher, Serif Ltd).

Manuscript 4: Research Paper (under peer-review)

Wyonna Rindt, Melanie Krug, Shuntaro Yamada, Franziska Sennefelder, Louisa Belz, Wen-Hui Cheng, Azeem Muhammad, Martin Kuric (MK), Marietheres Evers, Ellen Leich, Tanja Nicole Hartmann, Ana Rita Pereira, Marietta Hermann, Jan Hansmann, Mohammed Ahmed Yassin, Kamal Mustafa, Regina Ebert, and Franziska Jundt (2024): A 3D bioreactor model to study osteocyte differentiation and mechanobiology under perfusion and compressive mechanical loading, **Acta Biomaterialia**

Figure	**Author Initials,** Responsibility decreasing from left to right				
1	Wyonna Rindt				MK
2	Wyonna Rindt				
3	Wyonna Rindt				
4	Wyonna Rindt				
5	Wyonna Rindt				MK
6	Wyonna Rindt				MK
7	Wyonna Rindt				MK

Explanations: Martin Kuric contributed by counseling on experimental procedures and data analysis, such as quantifying normalized fluorescence intensity of immunohistochemistry and qPCR.

Manuscript 5: Research Paper (under revision)

Marietta Herrmann, Jutta Schneidereit, Susanne Wiesner, Martin Kuric (MK), Maximilian Rudert, Martin Lüdemann, Mugdha Srivastava, Norbert Schütze, Regina Ebert, Denitsa Docheva, Franz Jakob (2024): Peripheral blood cells enriched by adhesion to CYR61 are heterogenous myeloid modulators of tissue regeneration with early endothelial progenitor characteristics, **European Cells and Materials**

Figure	**Author Initials,** Responsibility decreasing from left to right				
1	Marietta Herrmann				
2	Marietta Herrmann				
3	Marietta Herrmann				
4	Marietta Herrmann				
5	Marietta Herrmann				
6	Marietta Herrmann				
7	Marietta Herrmann				MK

Explanations: Martin Kuric scanned osteogenically differentiated MSCs in Fig. 7 for quantification of alizarin red staining.

Manuscript 6: Research Letter (published)

Marietheres Evers, Martin Schreder, Thorsten Stühmer, Franziska Jundt, Regina Ebert, Tanja Nicole Hartmann, Michael Altenbuchinger, Martina Rudelius, Martin Kuric (MK), Wyonna Darleen Rindt, Torsten Steinbrunn, Christian Langer, Sofia Catalina Heredia-Guerrero, Hermann Einsele, Ralf Christian Bargou, Andreas Rosenwald, Ellen Leich (2023): Prognostic value of extracellular matrix gene mutations and expression in multiple myeloma, **Blood Cancer J. 13(1):43**

Figure	**Author Initials,** Responsibility decreasing from left to right				
1	Marietheres Evers				
2	Marietheres Evers				

Explanations: Martin Kuric contributed indirectly through counseling and feedback on submitted manuscript.

I also confirm my primary supervisor's acceptance.

Doctoral Researcher's Name Date Place Signature

D.3 Affidavit

Affidavit

I hereby confirm that my thesis entitled "Development and Semi-Automated Analysis of an in vitro Model for Myeloma Cells Interacting with Mesenchymal Stromal Cells" is the result of my own work. I did not receive any help or support from commercial consultants. All sources and / or materials applied are listed and specified in the thesis.

Furthermore, I confirm that this thesis has not yet been submitted as part of another examination process neither in identical nor in similar form.

Würzburg
Place, Date Signature

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, die Dissertation "Entwicklung und semi-automatisierte Analyse eines in vitro-Modells für Myelomzellen in Interaktion mit mesenchymalen Stromazellen" eigenständig, d.h. insbesondere selbständig und ohne Hilfe eines kommerziellen Promotionsberaters, angefertigt und keine anderen als die von mir angegebenen Quellen und Hilfsmittel verwendet zu haben.

Ich erkläre außerdem, dass die Dissertation weder in gleicher noch in ähnlicher Form bereits in einem anderen Prüfungsverfahren vorgelegen hat.

Würzburg
Ort, Datum Unterschrift

D.4 Curriculum Vitae

MARTIN KURIC

Cell Biologist | Data Scientist

97072 Würzburg
+49 15773816443
martin.kur4@gmail.com
martin-kuric
github.com/markur4

WHO AM I?

As a cancer cell biologist with a strong passion for data analysis and machine learning, I am seeking a position where I can utilize my creativity to automate tasks, solve complex problems or handle big data.

Microscopy	100%	Python	100%
Cancer Biology	90%	Image Analysis	90%
Stem Cells	80%	Statistics	80%
RNA Biochemistry	75%	ML/AI	75%

SELECTED PROJECTS

2024 **Python Software *"ImageP"*** **GitHub Repository**
Accelerates batch processing of images of different sizes and types by >100%.
`numpy / skimage / scipy`

2020-2024 **Python Software *"plotastic"*** **Journal of Open Source Software | GitHub Repository**
Published a statistical library that self-configures based on intuitive plotting parameters.
`pandas / matplotlib / pingouin / seaborn`

2018-2024 **Cancer Research Project** **Journal: Cancer Research Communications**
Worked in a team with up to three technical assistants and published a list of genes with relevance for survival of myeloma patients (under peer-review).
`Time-Lapse Microscopy / RNAseq / Analysis of Patient Survival`

26.05.2022 **Deep-Learning Assisted Image Cytometry** **Poster at "Achilles Conference"**
Measurement of per-cell parameters from large automated microscopy scans.
`Convolutional Neural Networks / Image Segmentation`

EDUCATION

28.01.2019 – 2024 **Dr. rer. nat. in Biomedicine** **Prof. Dr. Regina Ebert | University of Würzburg**
Research focus: Dissemination of multiple myeloma & mesenchymal stromal cell interactions

01.04.2017 – 2024 (parallel to M.Sc. & PhD) **Elite Biological Physics** **University of Bayreuth**
Interdisciplinary & international study program for exceptional students of physics or biology.

01.10.15 – 15.08.18 **M.Sc. in Biochemistry & Molecular Biology** **Prof. Dr. Claus-D. Kuhn | University of Bayreuth**
Research focus: RNA biochemistry, small RNAseq, stem cells & piRNAs in *S. mediterranea*

01.10.12 – 14.12.15 **B.Sc. in Biochemistry** **Prof. Dr. Benedikt Westermann | University of Bayreuth**
Research focus: Cell biology, mitochondrial inheritance in *S. cerevisiae*

LANGUAGES

German, English - C2
Slovakian - passive
French, Spanish - A2

SOFT SKILLS

Quality Management
Project Management
Violent Free Communication

HOBBIES

Coding - Python
Music - Piano & Guitar
Gym - Lift. Grow. Repeat

Würzburg — Location
05.03.2024 — Date
Martin Kuric — Signature