Course 62444
"Data Visualization and Analysis - Project"

Bashar Khaled Bdewi

s183356@student.dtu.dk

June 27, 2022

The final report

1. Seminar 1: Python/R and LATEX Tools and Platforms Review
   - Verifying R/RStudio
   - Verifying python spyder
   - Verifying the folder structure
2. Seminar 2: Data visualization and Analysis using Python/R Libraries on Laptop and Cloud
   - Data Visualization using R/RStudio
   - Vattenfall dataset visualization and analysis using R
   - Data Visualization using Python_Spyder
   - Data Visualization using python_Google CoLab
3. Seminar 3: Final Presentation
   - Data visualization in R/RStudio
   - Data visualization in Python
   - Data visualization in Julia

Seminar 1 - Table of Contents

1. **R/RStudio** enviroment
   The examples are from [Kabacoff, 2020]
2. **Python**/Spyder/ Jupyter Notebook enviroment
   The examples are from [VanderPlas, 2016]
3. **The LaTeX2e** Beamer Class
   Overleaf is used in this project.
4. The folder structure

Verifying R/RStudio

Verifying the R/RStudio environment using examples from [Kabacoff, 2020].

We nedd to get current working directory :

Session $\rightarrow$ Set Working Directory $\rightarrow$ 62444_PyR directory

"salaries.csv" dataset.

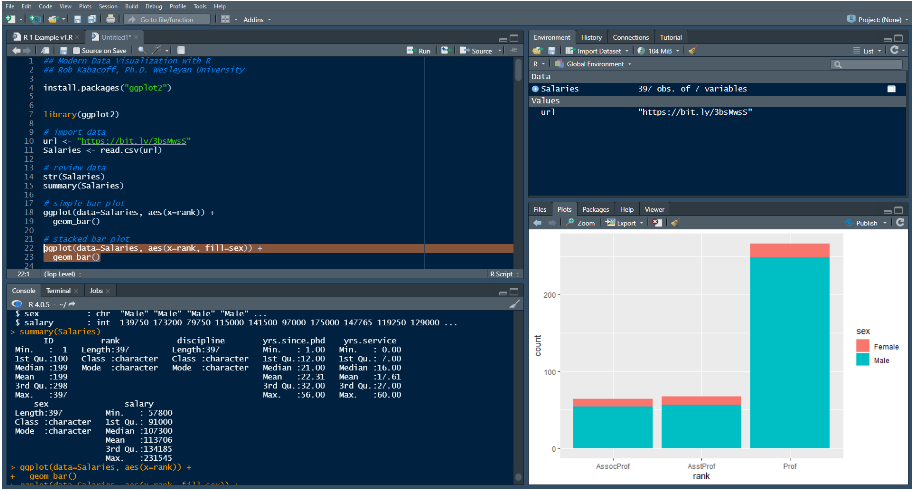
Reading the dataset:
Salaries <- read_csv ("salaries.csv")

Verifying Rstudio: ggplot...Stacked bar plot

Example

```
install.packages("ggplot2")
library(ggplot2)
# import data
url <- "https://bit.ly/3bsMwsS"
Salaries <- read.csv(url)
# review data
str(Salaries)
summary(Salaries)

# stacked bar plot
ggplot(data=Salaries, aes(x=rank, fill=sex)) +
  geom_bar()
```

Verifying Rstudio: ggplot...Stacked bar plot

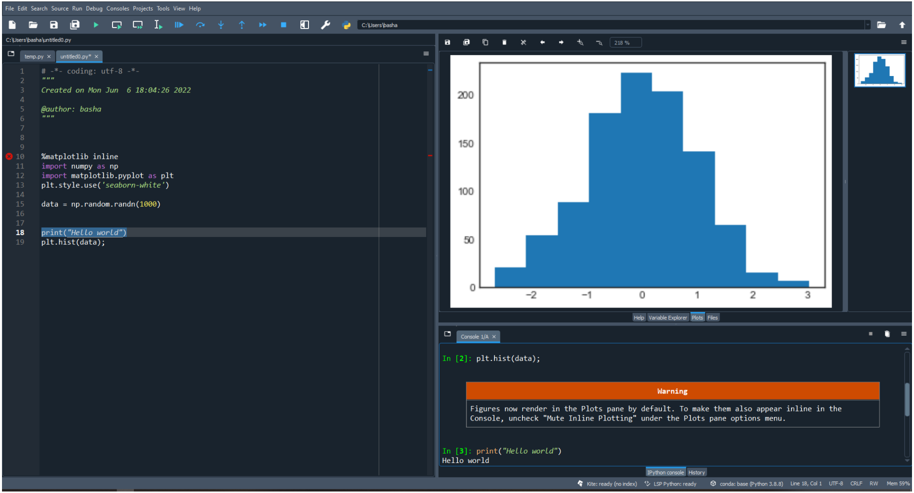


Verifying python spyder: Matplotlib...A simple histogram

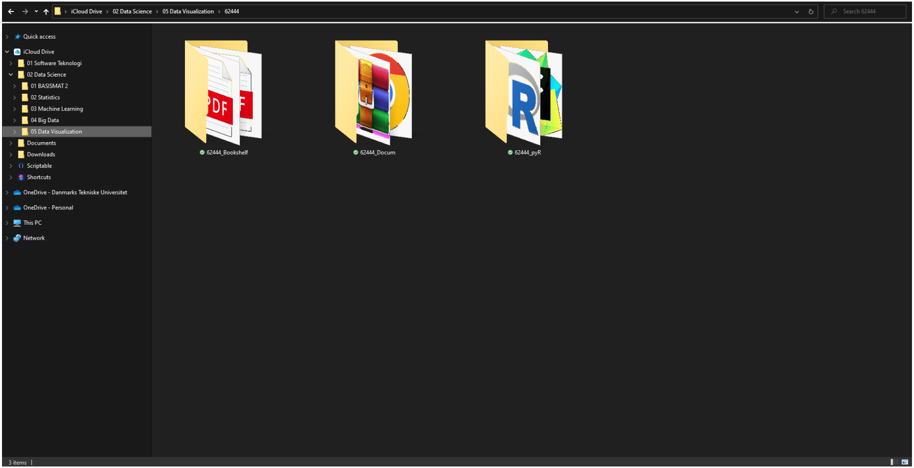
Example

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('seaborn-white')

data = np.random.randn(1000)
plt.hist(data);
```

Verifying python spyder: Matplotlib...A simple histogram



Verifying the working folder structure



Seminar 2 - Table of Contents

Seminar 2 - An example using for loop in R to count the number of even numbers in a vector.

Example (1.1)

```
x <- c(2,5,3,9,8,11,6)
count <- 0
for (val in x) {
if(val %% 2 == 0)
count = count+1
}
print(count)
```

Example (1.1 output:)

```
[1] 3
```

Seminar 2 - Using if-else-statement in R

Example (1.2)

```
x <- -5
if(x > 0){
print("Non-negative number")
} else {
print("Negative number")
}

# Output:
[1] "Negative number"
```

Seminar 2 - Using while loop in R to calculate factorial of a number

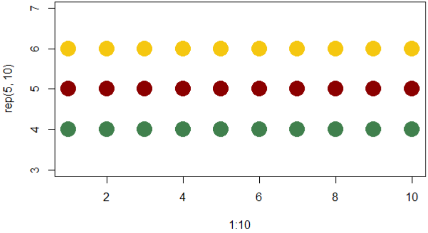
Example (1.3)

```
n <- 5

factorial <- 1
i <- 1

while (i <= n)
{
  factorial = factorial * i
  i = i + 1
}
print(factorial)
}

# Output: [1] 120
```

Example 2.1: Colors

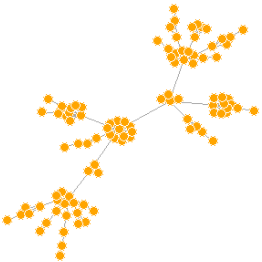
In most R functions, we can use named colors, hex, or rgb values, output of this lines of codes is showing the result:

```
plot(x=1:10, y=rep(5,10), pch=19, cex=5, col="#FFFF00")
points(x=1:10, y=rep(6, 10), pch=19, cex=5, col="pink")
points(x=1:10, y=rep(4, 10), pch=19, cex=5, col=rgb(.255, .0, .255))
```



Example 2.2: Network layouts

Network layouts are algorithms that return coordinates for each node in a network.
sample_pa() function is used to generate a simple graph starting from one node and adding more nodes and links based on a preset level of preferential attachment (Barabasi-Albert model)



Example (2.2: Network layouts code)

```
library(igraph)
net.bg <- sample_pa(100)
V(net.bg)$size <- 8
V(net.bg)$frame.color <- "white"
V(net.bg)$color <- "orange"
V(net.bg)$label <- ""
E(net.bg)$arrow.mode <- 0
plot(net.bg)
```

Seminar 2 - A Selection of Visualizations in R

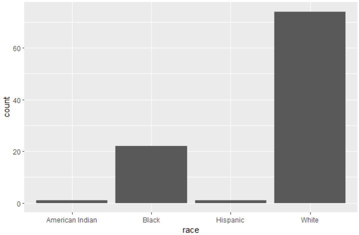
[Kabacoff, 2020] is used for preparing an R script which demonstrates the following types of visualizations:

- Categorical Data
  - Bar Charts
  - Pie Charts
- Distributions
  - Box Plots for Groups
- Times Series
- Scatter Plot

Example 3.1: Bar Chart

The Marriage dataset contains the marriage records of 98 individuals in Mobile County, Alabama. Below, a bar chart is used to display the distribution of wedding participants by race.

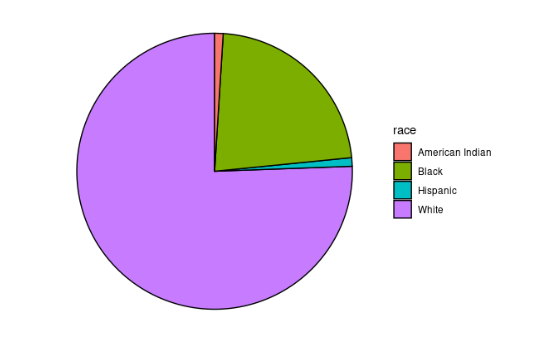
```
library(ggplot2)
data(Marriage, package = "mosaicData")
# plot the distribution of race
ggplot(Marriage, aes(x = race)) +
  geom_bar()
```



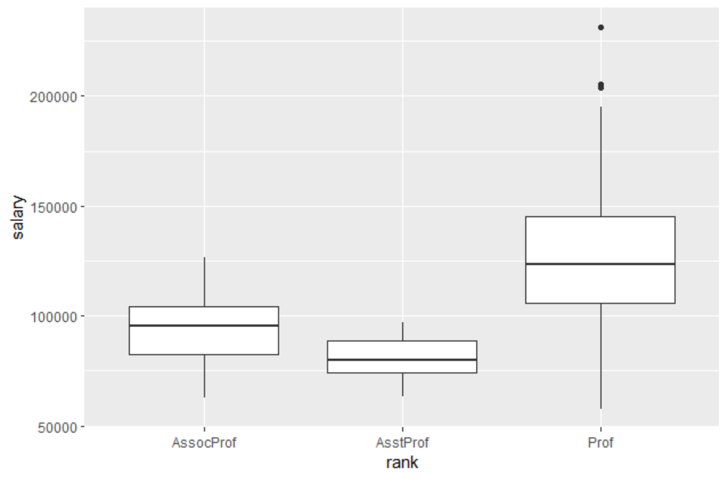
Seminar 2 - Categorical Data ... Pie Chart

Example (3.2: Pie Chart)

```r
# create a basic ggplot2 pie chart
library(dplyr)
plotdata <- Marriage %>%
  count(race) %>%
  arrange(desc(race)) %>%
  mutate(prop = round(n * 100 / sum(n), 1),
         lab.ypos = cumsum(prop) - 0.5  *prop)

ggplot(plotdata,
       aes(x = "",
           y = prop,
           fill = race)) +
  geom_bar(width = 1,
           stat = "identity",
           color = "black") +
  coord_polar("y",
              start = 0,
              direction = -1) +
  theme_void()
```



Example (3.4: Box plots)



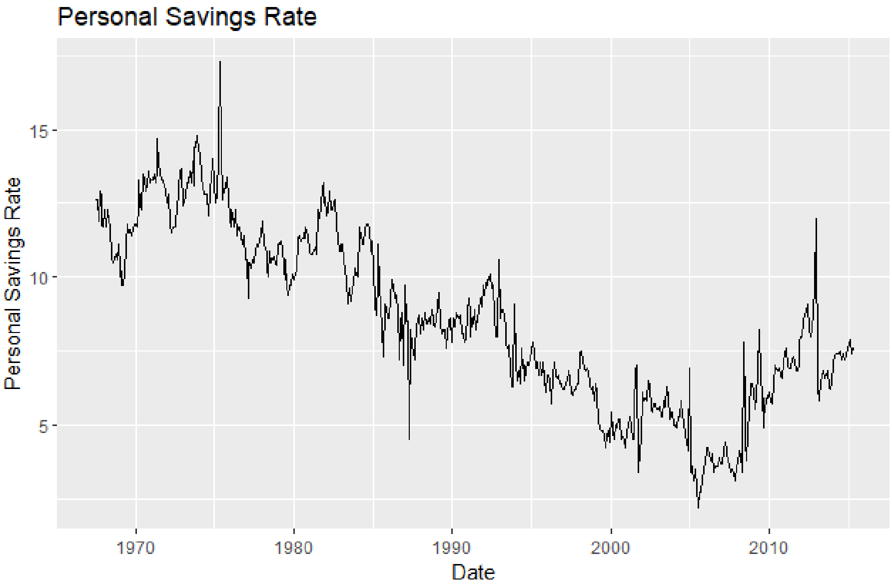
Seminar 2 - Times Series

A time series is a set of quantitative values obtained at successive time points. The intervals between time points (e.g., hours, days, weeks, months, or years) are usually equal.
Consider the Economics time series that come with the ggplot2 package. It contains US monthly economic data collected from January 1967 thru January 2015. Let's plot personal savings rate (psavert). We can do this with a simple line plot.

Example (3.5: Time Series)

```
library(ggplot2)
ggplot(economics, aes(x = date, y = psavert)) +
geom_line() +
labs(title = "Personal Savings Rate",
x = "Date",
y = "Personal Savings Rate")
```

Output

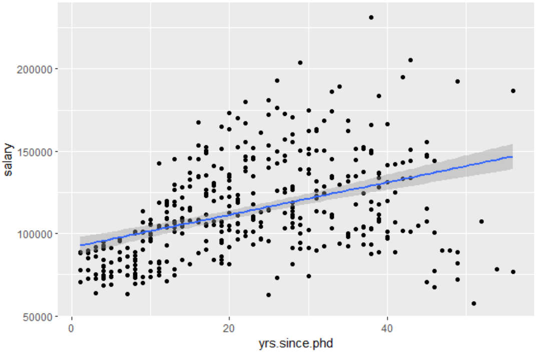


Personal Savings Rate
Personal Savings Rate
15
10
5
1970
1980
1990
2000
2010
Date

Seminar 2 - Quantitative vs Quantitative data ... Scatter Plot

Example (3.5: scatter plot with line of best fit)

ggplot(data=Salaries, aes(x=yrs.since.phd, y=salary)) +
geom_point() +
geom_smooth(method="lm", formula=y x)



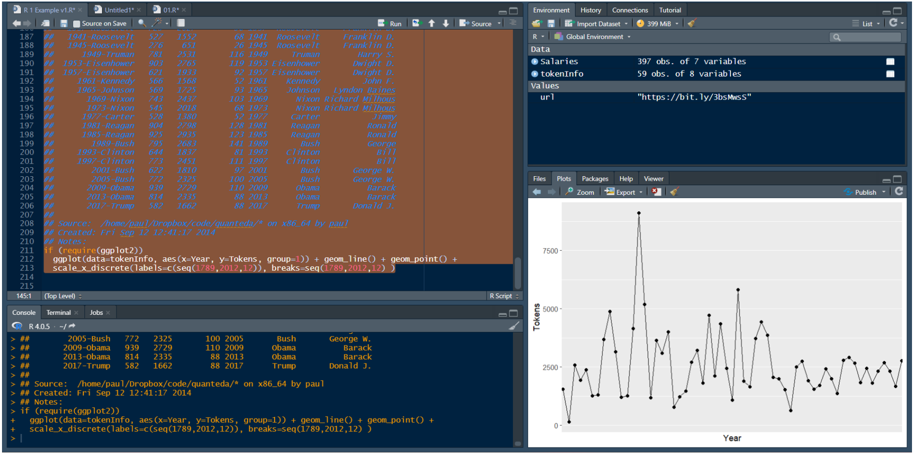
Seminar 2 - Text Analysis and Visualization in R ... Quanteda package

[Benoit, 2022] is used to verify the function of R-script for textual analysis.

Quanteda is an R package for managing and analyzing textual data developed.

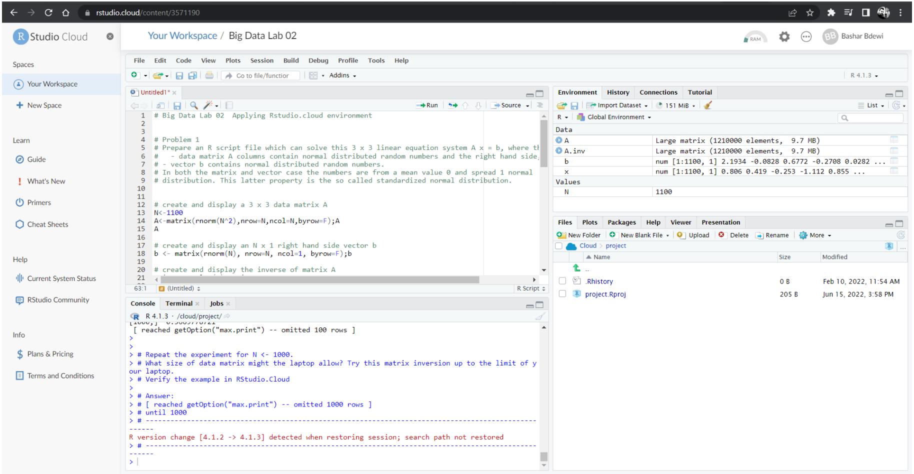
Quanteda makes it easy to manage texts in the form of a corpus, defined as a collection of texts that includes document-level variables specific to each text, as well as meta-data.

Quanteda R package



Seminar 2 - RStudio Cloud

Example of how some of the R scripts developed can run on RStudio.Cloud.



Seminar 2 - Vattenfall dataset visualization and analysis using R

We need to know what type of variables we are working with to choose the right statistical test for our data and interpret the results.

.

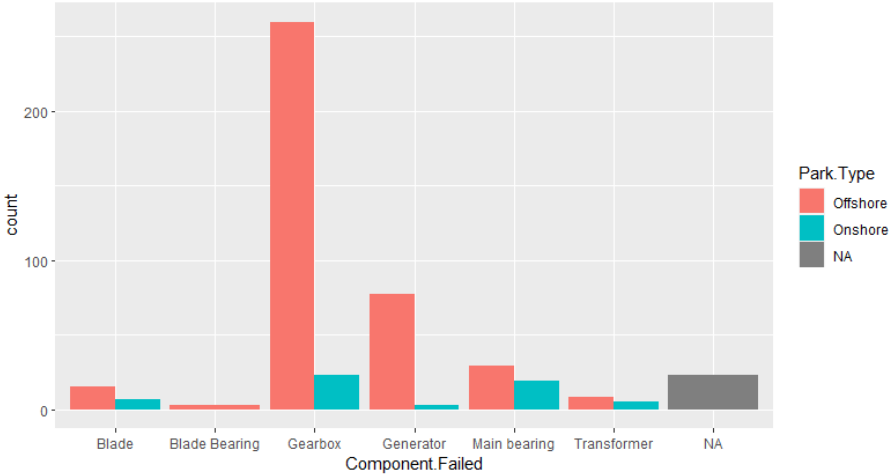
In Vattenfall dataset all variables except "Turbine Stop Date" and "Component Exchange Date" are categorical data.

Vattenfall dataset Visualization_Grouped bar plot

Example (Grouped bar plot)

```
# grouped bar plot
ggplot(data=wtf.df, aes(x=Component.Failed, fill=Park.Type)) +
geom_bar(position="dodge")
```

Grouped bar plot_"Component Failed" Vs. "Park Type"



Grouped bar plot_"Component Failed" Vs. "Park Type" (Graph Analysis)

When plotting the relationship between two categorical variables, stacked, grouped, or segmented bar charts are typically used. Here is example of grouped bar charts, that place bars for the second categorical variable side-by-side.
To create a grouped bar plot we use the position = "dodge" option.
We plotted plot the relationship between "Component Failed" "Park Type"

We can see here that the Gearbox is the most risky component and the offshore is the most difficult place.

- **for-loop**

```python
# Program to find the sum of all numbers stored in a list

# List of numbers
numbers = [6, 5, 3, 8, 4, 2, 5, 4, 11]

# variable to store the sum
sum = 0

# iterate over the list
for val in numbers:
    sum = sum+val

print("The sum is", sum)

```

output: The sum is 48

- **if-statement**

```python

# If the number is positive, we print an appropriate message

num = 3
if num > 0:
    print(num, "is a positive number.")
print("This is always printed.")

num = -1
if num > 0:
    print(num, "is a positive number.")
print("This is also always printed.")
```

output :
3 is a positive number.
This is always printed.
This is also always printed.

Seminar 2 - Python Language elements

- **while-loop**

```python
'''Example to illustrate the use of else statement with the while loop'''

counter = 0

while counter < 3:
    print("Inside loop")
    counter = counter + 1
else:
    print("Inside else")
```

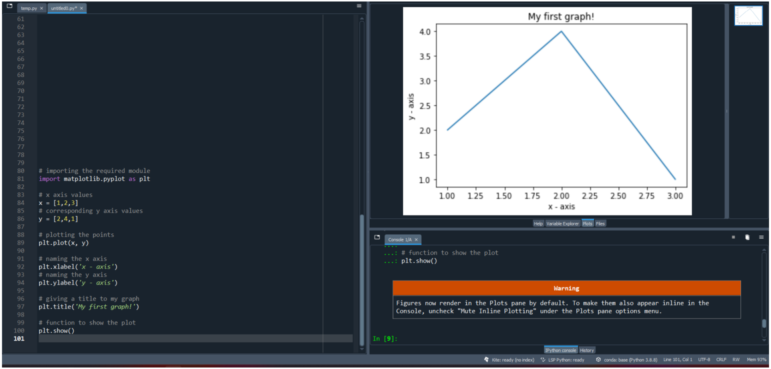
output:
Inside loop Inside loop
Inside loop
Inside else

- **Python function call**

```python

 # basic example of subtractig 2 numbers
def subtractNum():
    print(34 - 4)

subtractNum()
```

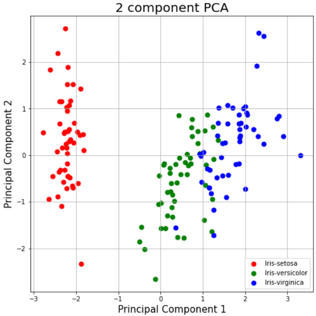
output: 30

- **matplotlib:**
  Collection of functions that make matplotlib work like MATLAB. Each pyplot function makes a change to a figure: creates a figure, creates a plotting area in a figure, decorates the plot with labels, etc.

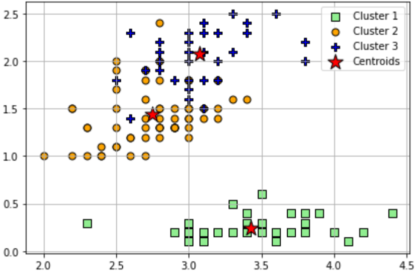


- **Scikit-Learn:**
  Machine learning library, uses NumPy for high-performance linear algebra and array operations. Main tool areas/algorithms:

  - Dimensionality Reduction: Techniques for reducing the number of input variables in training data. In high dimensional data, useful to reduce the dimensionality by projecting the data to a lower dimensional subspace which captures the "essence" of the data. Unsupervised machine learning.

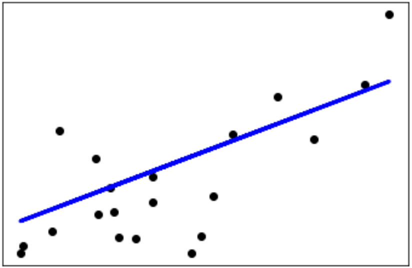


- Clustering:
  Method of identifying and grouping similar data points in larger datasets.
  Used to classify data into structures that are more easily understood and manipulated. Examples are unlabeled, unsupervised machine learning. Mean shift clustering: a centroid-based algorithm, which works by updating candidates for centroids to be the mean of the points within a given region.



- Regression:

  Statistical method for modelling relationship between a dependent variable with a given set of independent variables.



- **Numpy:**
  Numerical Python, for working with arrays. Facilitate advanced mathematical operations on large numbers of data.
  Array objects that are 50x faster than traditional Python lists.

```
# create a numpy ndarray object
#-------------------------------------------------------------------------------#
import numpy as np

arr = np.array([1, 2, 3, 4, 5])

print(arr)

print(type(arr))
```

- **Plotly:**
  For interactive, publication-quality graphs.Supports over 40 unique chart types covering a wide range of statistical, financial, geographic, scientific, and 3-dimensional use-cases.

```
pip install plotly==5.8.2

import plotly.express as px
fig = px.bar(x=["a", "b", "c"], y=[1, 3, 2])
fig.write_html('first_figure.html', auto_open=True)
```

- **Pandas:**
  Provides several different options for visualizing your data with .plot()
  Here it extracts the data from the .csv file into a Dataframe:

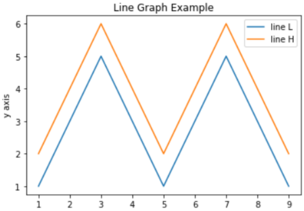
```python
# Pandas
#-----------------------------------------------------------------------
import pandas as pd

df = pd.read_csv('data.csv')

print(df.to_string())

```

Seminar 2 - Google CoLab

visualization functions:

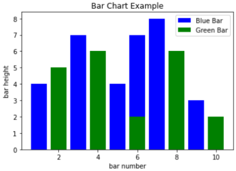
- line plots

```
#line plots
import matplotlib.pyplot as plt

x  = [1, 2, 3, 4, 5, 6, 7, 8, 9]
y1 = [1, 3, 5, 3, 1, 3, 5, 3, 1]
y2 = [2, 4, 6, 4, 2, 4, 6, 4, 2]
plt.plot(x, y1, label="line L")
plt.plot(x, y2, label="line H")
plt.plot()

plt.xlabel("x axis")
plt.ylabel("y axis")
plt.title("Line Graph Example")
plt.legend()
plt.show()
```



- bar plots

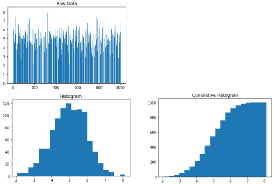
```
import matplotlib.pyplot as plt

# Look at index 4 and 6, which demonstrate overlapping cases.
x1 = [1, 3, 4, 5, 6, 7, 9]
y1 = [4, 7, 2, 4, 7, 8, 3]

x2 = [2, 4, 6, 8, 10]
y2 = [5, 6, 2, 6, 2]

# Colors: https://matplotlib.org/api/colors_api.html

plt.bar(x1, y1, label="Blue Bar", color='b')
plt.bar(x2, y2, label="Green Bar", color='g')
plt.plot()

plt.xlabel("bar number")
plt.ylabel("bar height")
plt.title("Bar Chart Example")
plt.legend()
plt.show()
```



visualization functions:

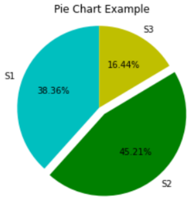
- histograms

```python
import matplotlib.pyplot as plt
import numpy as np

# Use numpy to generate a bunch of random data in a bell curve around 5.
n = 5 + np.random.randn(1000)

m = [m for m in range(len(n))]
plt.bar(m, n)
plt.title("Raw Data")
plt.show()

plt.hist(n, bins=20)
plt.title("Histogram")
plt.show()

plt.hist(n, cumulative=True, bins=20)
plt.title("Cumulative Histogram")
plt.show()
```



- pie chart

```python
import matplotlib.pyplot as plt

labels = 'S1', 'S2', 'S3'
sections = [56, 66, 24]
colors = ['c', 'g', 'y']

plt.pie(sections, labels=labels, colors=colors,
        startangle=90,
        explode = (0, 0.1, 0),
        autopct = '%1.2f%%')

plt.axis('equal') # Try commenting this out.
plt.title('Pie Chart Example')
plt.show()
```

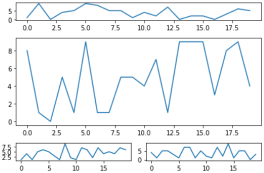


Google CoLab

visualization functions:

- subplot

```python
import matplotlib.pyplot as plt
import numpy as np

def random_plots():
  xs = []
  ys = []

  for i in range(20):
    x = i
    y = np.random.randint(10)

    xs.append(x)
    ys.append(y)

  return xs, ys

fig = plt.figure()
ax1 = plt.subplot2grid((5, 2), (0, 0), rowspan=1, colspan=2)
ax2 = plt.subplot2grid((5, 2), (1, 0), rowspan=3, colspan=2)
ax3 = plt.subplot2grid((5, 2), (4, 0), rowspan=1, colspan=1)
ax4 = plt.subplot2grid((5, 2), (4, 1), rowspan=1, colspan=1)

x, y = random_plots()
ax1.plot(x, y)

x, y = random_plots()
ax2.plot(x, y)

x, y = random_plots()
ax3.plot(x, y)
x, y = random_plots()
ax4.plot(x, y)

plt.tight_layout()
plt.show()
```

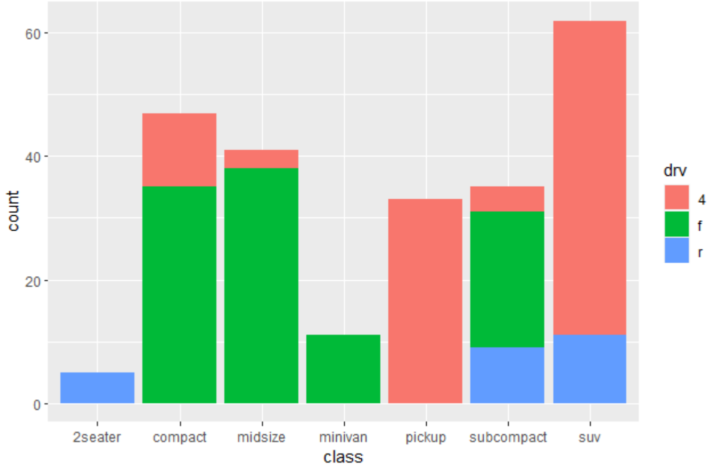


Seminar 3 - Table of Contents

Categorical vs. Categorical

Example (stacked bar chart)

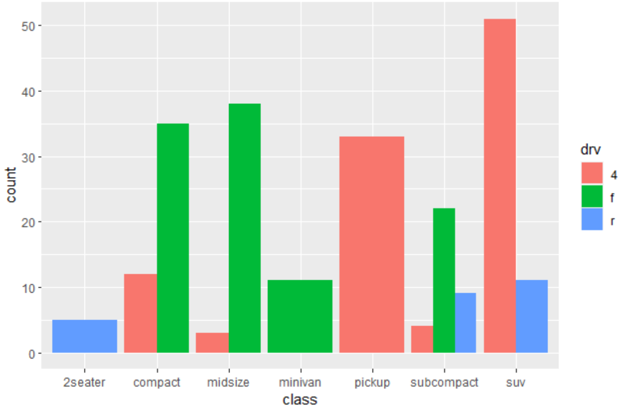
ggplot(mpg, aes(x = class, fill = drv)) + geom$_b$*ar*(*position* = "*stack*)



Categorical vs. Categorical

Example (grouped bar plot)

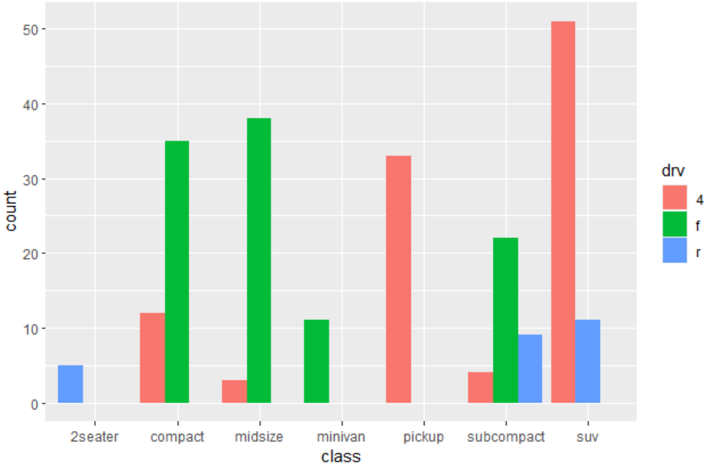
Grouped bar charts place bars for the second categorical variable side-by-side. To create a grouped bar plot use the position = "dodge" option.



Categorical vs. Categorical

Example (grouped bar plot preserving zero count bars)

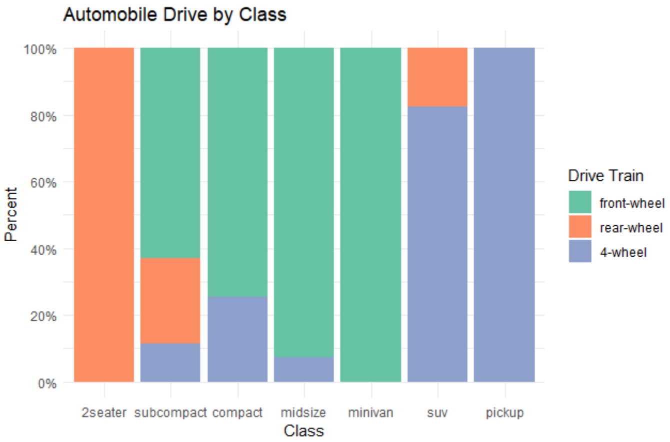
Side-by-side bar chart with zero count bars retained



Categorical vs. Categorical

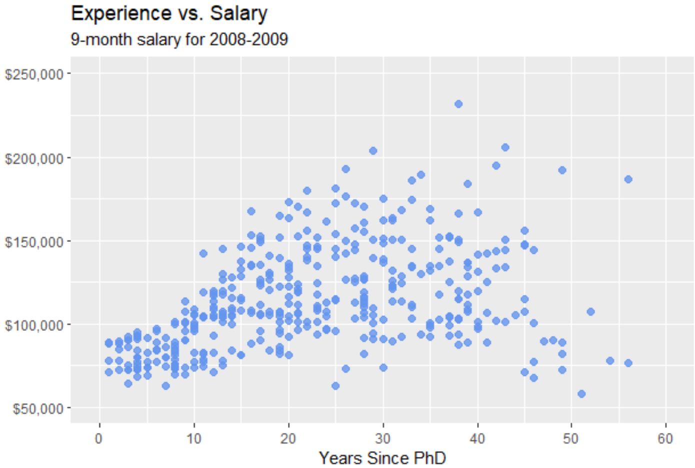
Example (bar plot, with each bar representing 100reordered bars, and better labels and colors)

Segmented bar chart with improved labeling and color



Example (enhanced scatter plot)

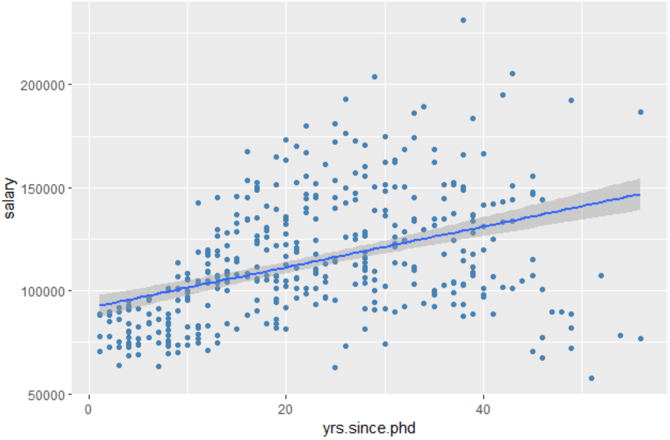
Scatterplot with color, transparency, and axis scaling



Quantitative vs. Quantitative

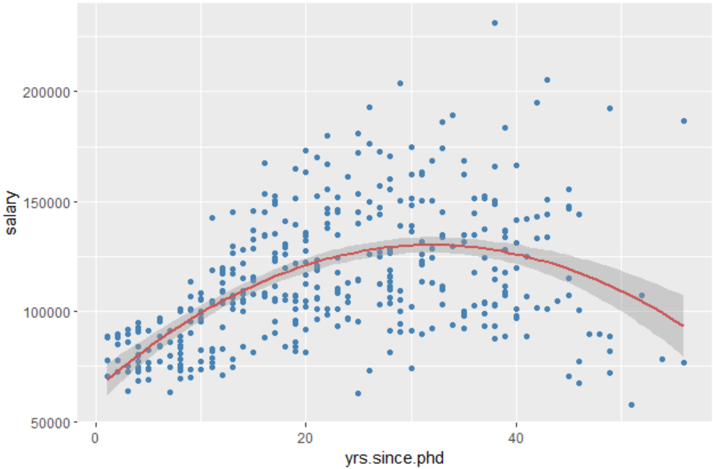
Example (scatterplot with linear fit line)

It is often useful to summarize the relationship displayed in the scatterplot, using a best fit line.



Example (scatterplot with quadratic line of best fit)

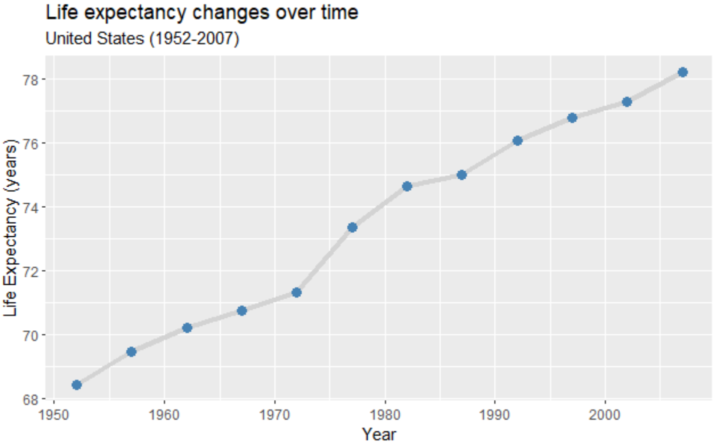
Applying a quadratic fit to the salary dataset produces the following result



Categorical vs. Categorical

Example (line plot with points)

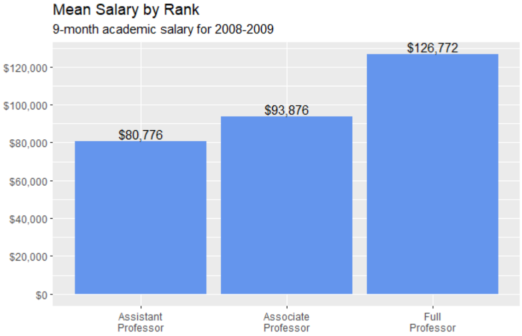
When one of the two variables represents time, a line plot can be an effective method of displaying relationship.we'll add points as well.



Categorical vs. Quantitative

Example (plot mean salaries in a more attractive fashion)

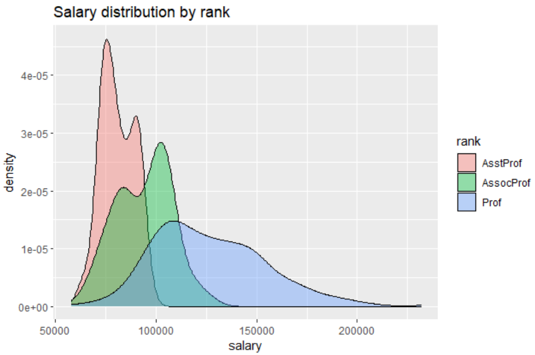
We can make it more attractive with some options. One limitation of such plots is that they do not display the distribution of the data - only the summary statistic for each group. Grouped kernel density plots correct this limitation to some extent.



Categorical vs. Quantitative

Example (plot the distribution of salaries by rank using kernel density plots)

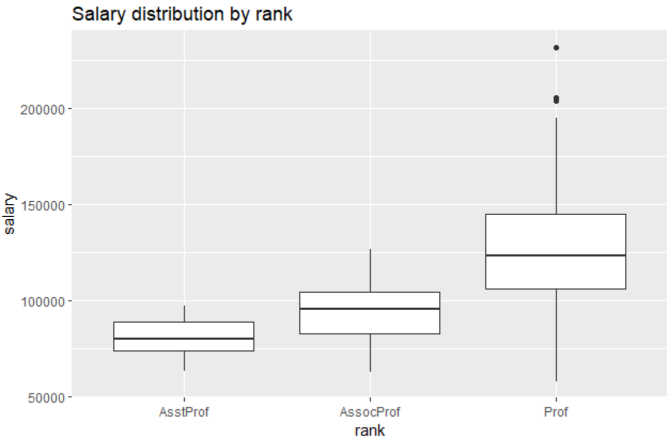
compare groups on a numeric variable by superimposing kernel density plots in a single graph. The graph makes clear that, in general, salary goes up with rank. However, the salary range for full professors is very wide



Categorical vs. Quantitative

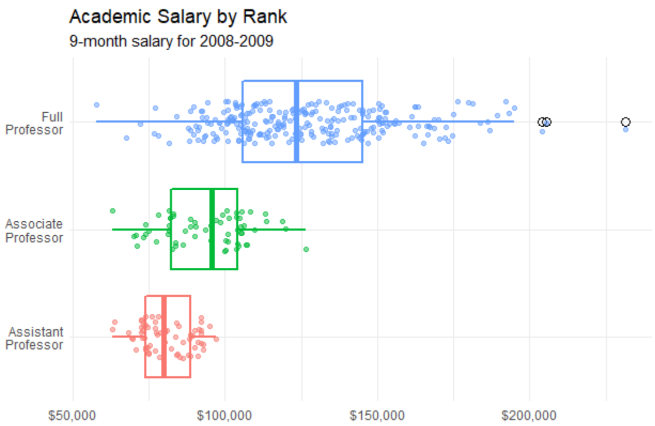
Example (plot the distribution of salaries by rank using boxplots)

Side-by-side box plots are very useful for comparing groups (i.e., the levels of a categorical variable) on a numerical variable.



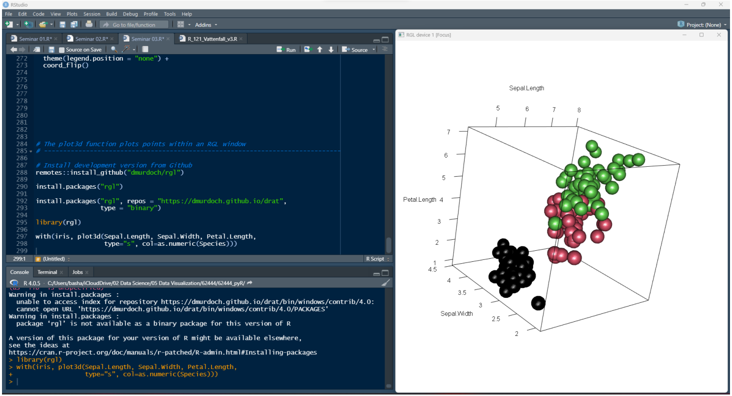
Example (plot the distribution of salaries by rank using jittering)

It may be easier to visualize distributions if we add boxplots to the jitter plots.



Example (he plot3d function plots points within an RGL window)

The rgl package is used to produce interactive 3-D plots.The plot3d function plots points within an RGL window. It is similar to the classic plot function, but works in 3 dimensions.



Seminar 3 - The difference between categorical and quantitative data:

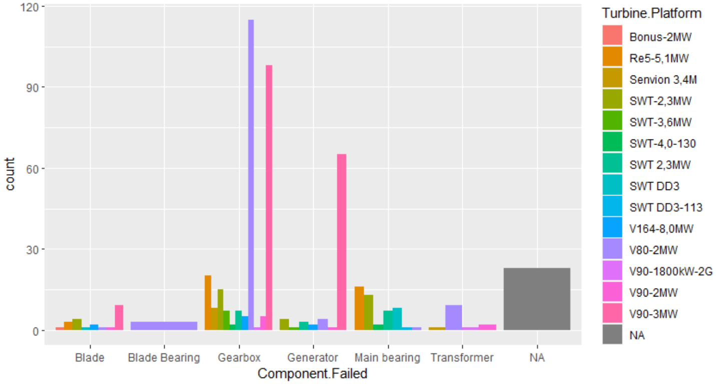
Quantitative Quantitative data is defined as the value of data in the form of counts or numbers, that represents amounts like weight, height and age.

Categorical variables is data which represents groups, like race, sex.

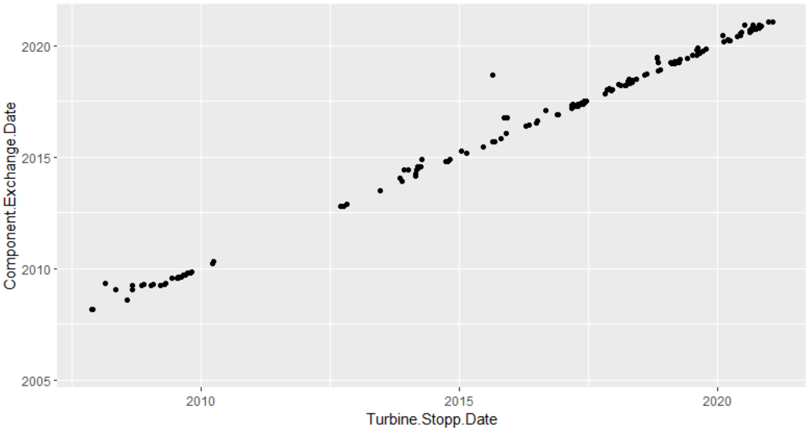
We need to know what type of variables we are working with to choose the right statistical test for our data and interpret the results. e.g. in Vattenfall dataset all variables except "Turbine Stop Date" and "Component Exchange Date" are categorical data.

"Component Failed" Vs. "Turbin platform"

We can see here that the Gearbox is the most risky component and the offshore is the most difficult place. The following graph shows the distribution of Turbine Platform in different component failures, it shows that the most common failed is the V80-2MW.



"Turbine Stop Date" Vs. "Component Exchange Date"



A descriptive analysis of the Vattenfall data set

Descriptive statistics is the term given to the analysis of data that helps describe, show or summarize data in a meaningful way. By visualization of categorical and quantitative data on Vattenfall data set, we can conclude that there are more fails on the Gearbox component. The location will also effect the component failure, as the plot shows, the offshore is the most difficult place. The scatterplot shows that Component exchange will effect the life time of a turbine.

Encoding categorical features

Example (OrdinalEncoder)

Often features are not given as continuous values but categorical.To convert categorical features to such integer codes, we can use the OrdinalEncoder. This estimator transforms each categorical feature to one new feature of integers (0 to $n_categories - 1$)

```python
# OrdinalEncoder.
import numpy as np
import pandas as pd
from sklearn import preprocessing

from sklearn.preprocessing import OrdinalEncoder
enc = preprocessing.OrdinalEncoder()
X = [['male', 'from US', 'uses Safari'], ['female', 'from Europe', 'uses Firefox']]
enc.fit(X)

enc.transform([['female', 'from US', 'uses Safari']])
```

Out: array([[0., 1., 1.]])

Encoding categorical features

Example (OneHotEncoder)

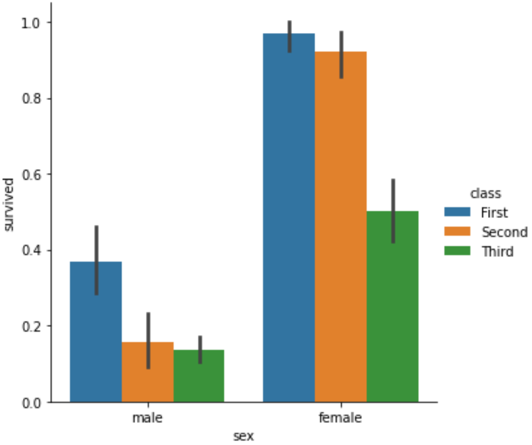
to convert categorical features to features that can be used with scikit-learn estimators is to use a one-of-K or dummy encoding. This encoding can be obtained with the OneHotEncoder.

```python
# OneHotEncoder.
from sklearn.preprocessing import OneHotEncoder
enc = preprocessing.OneHotEncoder()
X = [['male', 'from US', 'uses Safari'], ['female', 'from Europe', 'uses Firefox']]
enc.fit(X)
enc.transform([['female', 'from US', 'uses Safari'],
               ['male', 'from Europe', 'uses Safari']]).toarray()

```

Out: array([[1., 0., 0., 1., 0., 1.], [0., 1., 1., 0., 0., 1.]])

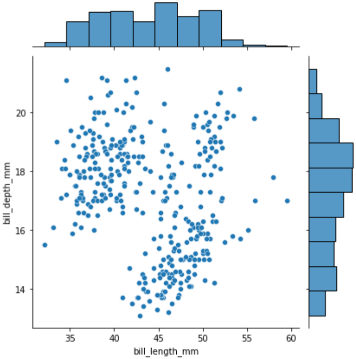
Example (Categorical vs Categorical)

Bar plots

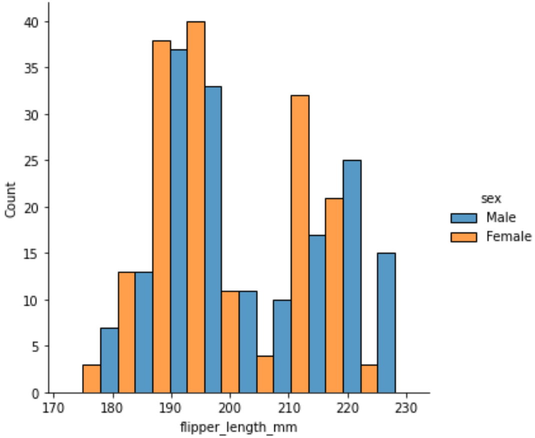


Example (quantitative vs quantitative)

Plotting joint and marginal distributions. jointplot(), which augments a bivariate relatonal or distribution plot with the marginal distributions of the two variables.

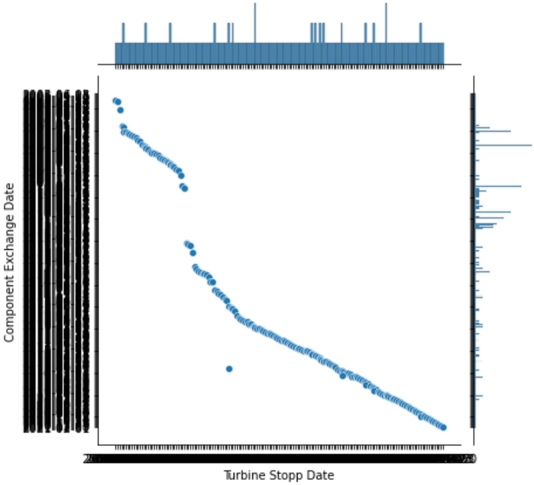


Example (quantitative vs categorical)



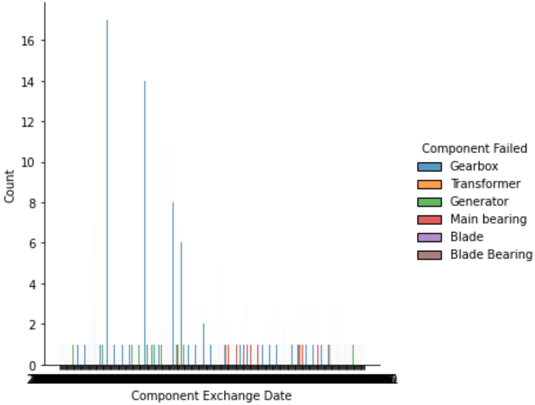
Vattenfall dataset visualization and analysis

Example ("Turbine Stopp Date" vs "Component Exchange Date")



Vattenfall dataset visualization and analysis

Example ("Component Exchange Date "vs hue="Component Failed")



Why the language Julia?

- Easy to use
- Free and open source
- Flexible dynamic language for high performance (fast)
- Brings high level dynamic and compiled languages together

Application areas:

- Appropriate for scientific and numerical computing
- For building entire Applications and Microservices

Julia Language:

- Has no classes / class-specific methods
- Standard Libraries and popular functions already included

References I

Anaconda Distribution (2022)
www.anaconda.com
www.anaconda.com/distribution

Kenneth Benoit (2022)
"Quantitative Analysis of Textual Data".
R-Package at CRAN 2022
https://cran.r-project.org/web/packages/quanteda/quanteda.pdf

Kenneth Benoit (2022)
"vignettes quanteda: Quick Start Guide". *R-Package at CRAN 2022*
https://cran.r-project.org/web/packages/quanteda/vignettes/quickstart.html

Google CoLab (2022)
https://colab.research.google.com

References II

- Reka Horvath (2020)
  "Plot With Pandas: Python Data Visualization for Beginners"
  https://realpython.com/pandas-plot-python/
- The Julia Language (2022)
  https://julialang.org/
- Jupyter Notebook (2021)
  "The Jupyter Notebook". *http://jupyter.org/*
- Robert Kabacoff (2020)
  "Data Visualization with R". *https://rkabacoff.github.io/datavis/*
- Sharon Machlis (2019)
  "Great R packages for data import, wrangling and visualization".
  Computerworld, 2019

References III

Wes McKinney (2022)
"pandas: powerful Python data analysis toolkit"
https://pandas.pydata.org/docs/pandas.pdf

Stephan Müller, Kenneth Banoit (2020)
"quanteda - Cheat Sheet".
https://www.rstudio.com/resources/cheatsheets/

Tobias Oetiker, Hubert Partl, Irene Hyna and Elisabeth Schlegl (2021)
"The Not So Short Introduction to LaTeX2e".
https://tobi.oetiker.ch/lshort/lshort.pdf

Katherine Ognyanova (2021)
"Network visualization with R".
https://kateto.net/network-visualization

References IV

The Python Package Index (2022)
"The Python Package Index".
https://pypi.org/

The R Project for Statistical Computing (2022)
https://www.r-project.org/

Min Ragan-Kelley, Carol Willing, Jason Grout (2018)
"Jupyter: Tools for the Life Cycle of a Computational Idea".
SIAM News Volume 51, Issue 2 March 2018

Hans Rosling (2018)
"Gabminder".
http://www.gapminder.org

RStudio Cheat Sheets (2022)
https://www.rstudio.com/resources/cheatsheets/

References IIV

Machine Learning Map index (2022)
https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

The seaborn library for graphics in Python (2022).
https://seaborn.pydata.org/

Louis Stewart, Joseph Wright (2022)
"beamer – A LaTeX class for producing presentations and slides".
https://github.com/josephwright/beamer

Till Tantau, Joseph Wright, Vedran Miletić (2022)
"The beamer class".
http://tug.ctan.org/macros/latex/contrib/beamer/doc/beameruserguide.pdf

Paul Torfs, Claudia Brauer (2014)
"A (very) short introduction to R".
https://cran.r-project.org/doc/contrib/Torfs+Brauer-Short-R-Intro.pdf

References IIV

Jake VanderPlas (2016)
"A Whirlwind Tour of Python".
https://github.com/jakevdp/WhirlwindTourOfPython

The End