Data Science Lessons Inspired by Kepler's Laws

An MAA Book Chapter

Boyan Kostadinov*

New York City College of Technology, CUNY

Contents

1 Computing Framework **1**

2 Kepler's Laws **2**

3 Discovering Kepler's 3rd Law **3**
3.1 Visualizing Planetary Data from NASA 3
3.2 Kepler's 3rd Law from Normalized Planetary Data 4
3.3 Kepler's 3rd Law from Raw Planetary Data 7

4 Discovering Kepler's 1st Law **11**
4.1 Kepler's Measurements of Mars 11
4.2 Fitting an Ellipse to Kepler's Mars Data 12

5 Discovering Kepler's 2nd Law **14**

6 Conclusions **17**

1 Computing Framework

We scaffold the projects by presenting step-by-step the mathematical ideas, along with the computational techniques needed for the students to work through all activities. We use the R programming language [7] for all computations and visualizations. We recommend using RStudio [1], either locally installed, or RStudio Cloud [2]. Prior programming experience in R is not required, but in this case the students are expected to learn the basics of R and R Markdown. RStudio supports R Markdown (or the new generation Quarto) notebooks, which allows one to unify plain text narrative, mathematical expressions in LaTeX, and R code (or Python via the `reticulate` R package), and create fully reproducible, publication quality project reports and presentations.

We recommend the following resources to learn the basics of R: [5, 8], and R Markdown: [22, 10, 33]. We use the `mosaic`, `mosaicCalc` and `ggformula` R packages, developed by Project MOSAIC, designed to facilitate the use of R in statistics and calculus instruction, [21, 14, 13]. We also use the `tidyverse` [30, 26] collection of packages developed by RStudio, and specifically `ggplot2` [29] for visualizations, and `dplyr` [28] for data analysis. The reader can find additional resources on computational-problem solving with R in [12, 4, 18].

We have implemented all computations and visualizations in this chapter using R in an R Markdown notebook (Rmd), which can be downloaded from GitHub [17]. The Rmd notebook can be run in RStudio, which is freely available for Windows, Mac, and Linux:

*Mathematics Department, bkostadinov@citytech.cuny.edu

- R installers: https://cloud.r-project.org/
- RStudio installers: https://www.rstudio.com/products/rstudio/download/

A cloud-based option (free and paid) for using RStudio is provided by RStudio Cloud, [2], which also offers an instructor's account for creating virtual classrooms. RStudio Cloud now offers a real-time collaboration feature for shared projects. RStudio Cloud also offers an R interface to Python via the `reticulate` R package, which provides a comprehensive set of tools for interoperability between Python and R [25]. Thus, one can use both R and Python code cells in the same R Markdown (or Quarto) document, combining the power of both. With R Markdown, one can even create websites, blogs and books, using the `blogdown` and `bookdown` packages. For more details, see the resources tab on the RStudio website [1].

2 Kepler's Laws

Johannes Kepler was the young assistant of the 16th century Danish astronomer Tycho Brahe (1546 - 1601), considered the father of modern astronomy. Brahe had the amazing skill of being able to measure planetary position, by naked eye and his well-designed instruments, with amazing accuracy, as the telescope was not invented until 1608.

Brahe made very accurate and comprehensive astronomical observations, but it took the mathematical genius of Kepler and two decades of hard work to fit Brahe's data to three simple empirical laws of planetary motion:

1. **First law**: Kepler discovered that the orbit of Mars was an ellipse and he generalized this observation that every planet has an elliptical orbit with the Sun at one of its foci. See Figure 1.
2. **Second law**: The radius vector from the Sun to a planet sweeps out equal areas in equal times, implying that the rate of change (with respect to time) of the area swept by the radius vector of the planet's orbit is constant.
3. **Third law**: The orbital period T of a planet revolving around the Sun is related to the average distance from the Sun, D of its elliptical orbit by $D^3 = KT^2$, where K is constant across all planets.

In 1609 Kepler published the first two laws of planetary motion in *Astronomia Nova* [23], where he described his discoveries and how he dealt with noisy data to make the scientific conclusions known as Kepler's laws. This was an early example of what we now call *the scientific method*. In 1619 he published *Harmonices Mundi* (The Harmony of the World) where he describes his "third law" of planetary motion [16].

Kepler's second law proved crucial to Isaac Newton (1643 - 1727), when he derived from empirical observations his universal law of gravitation, first published in his *Principia* in 1687. In fact, Newton showed that the motion of bodies subject to central gravitational force need not always follow elliptical orbits, but can also have parabolic or hyperbolic orbits, depending on the total energy of the body. Thus, an object with sufficiently high energy, such as a comet, can enter the solar system and leave without ever returning. For more details and a video by the physicist Brian Green demonstrating how Newton's law of gravitation determines the trajectories of the planets, we refer the reader to [6].

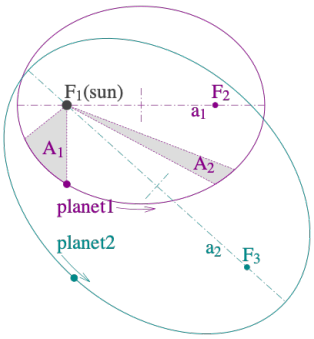

Figure 1: Kepler's laws for planetary motions. Source: Wikipedia.

We list below some important observations about Kepler's laws:

- Even a comet obeys Kepler's 3rd law, and from its period one can calculate the average distance of the comet to the Sun.
- Kepler's laws can be derived from Newton's laws of motion and his law of gravity.
- Newton's law of gravity can also be derived from Kepler's first two laws.
- Kepler's third law allows us to find the most efficient trajectory from Earth to Mars, which is called the Hohmann transfer orbit. Kepler's third law implies that this orbit has to be an ellipse that touches both orbits and has the shortest possible semimajor axis.

For more details on Kepler's laws, one can explore the freely available online Physics Bootcamp [19].

3 Discovering Kepler's 3rd Law

3.1 Visualizing Planetary Data from NASA

In this activity, we illustrate how *Kepler's Third Law of Planetary Motion* can be discovered using *least squares* to fit a mathematical model to real planetary data from NASA's Lunar and Planetary Science Division [32]. The planetary data in Table 1 were derived from the raw NASA data by doing data transformations that consist of normalizing the distances and the orbital periods in Earth's units (1AU $= 1.496 \times 10^8$ km) [31]. We give the raw planetary data from NASA in the code chunk below, and in the following code cell we normalize the data.

```
# Raw planetary data: distance is given in units of 10^6 km, and period in days
dist<-c(57.9,108.2,149.6,227.9,778.6,1433.5,2872.5,4495.1,5906.4) # in 10^6 km
period<-c(88,224.7,365.2,687,4331,10747,30589,59800,90560) # in Earth's days
```

Table 1: NASA's Normalized Planetary Data

	Distance D [AU]	Period T [Earth's Years]	ln(D)	ln(T)
Mercury	0.3870321	0.2409639	-0.9492477	-1.4231083
Venus	0.7232620	0.6152793	-0.3239837	-0.4856790
Earth	1.0000000	1.0000000	0.0000000	0.0000000
Mars	1.5233957	1.8811610	0.4209419	0.6318891
Jupiter	5.2045455	11.8592552	1.6495324	2.4731086
Saturn	9.5822193	29.4277108	2.2599092	3.3819368
Uranus	19.2012032	83.7595838	2.9549729	4.4279506
Neptune	30.0474599	163.7458927	3.4027781	5.0983158
Pluto	39.4812834	247.9737130	3.6758267	5.5133227

We can use the normalized planetary data to find a relationship between the average distance from the Sun, given in AU, and the orbital period, given in Earth's years. The code chunk below normalizes the planetary data, which is shown in Table 1. We can also visualize the normalized planetary data by creating a scatterplot of period vs. distance, see Exercise 3.1.

```
D<-dist/dist[3] # normalized distance in AU relative to Earth
T<-period/period[3] # normalized period in years relative to Earth
```

Exercise 3.1. Implement the two code cells above, plus the code chunk below in an R Markdown document in RStudio, and create a scatterplot of the normalized planetary data shown in Table 1. Note that the `ggplot()` function is from the `ggplot2` package, a part of the `tidyverse` collection. Also note that we customized the colors of the planets as well as their relative sizes, mostly for fun.

```
dataDT<-tibble(D,T) # column-bind D and T into a dataframe
colors<-c("black","brown","blue","red","orange","gold","lightblue","darkblue","black")
f <- 1.3 # a multiple to scale the planet sizes
```

```
sizes<-f*c(1/3,0.9,1,1/2,11,9,4,3.9,1/5) # relative planet sizes
ggplot() +
  geom_point(mapping=aes(x=D,y=T),data=dataDT,col=colors,size=sizes,alpha=0.6) +
  labs(x ="Distance D [AU]", y = "Orbital Period T [years]") +
  theme(axis.text=element_text(size=7), axis.title=element_text(size=8))
```

It is often a good idea to transform the data and visualize the relationship between the transformed data in our pursuit to discover any useful patterns. Many laws of nature are in the form of power laws, thus transforming the data to a log-log scale by taking logarithms of both distance and period, is a natural transformation to consider. The natural logarithms of distance and period are given in Table 1, and the log-log plot is shown in Figure 2.

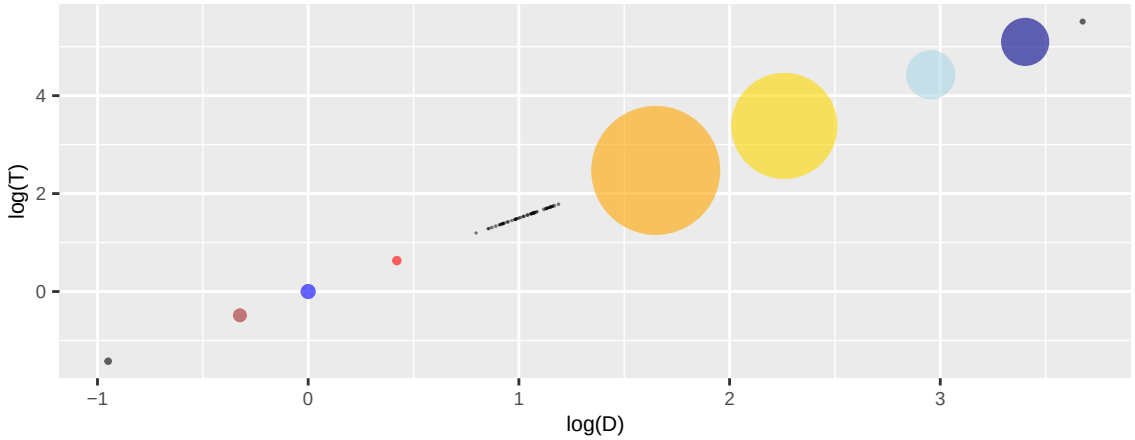

Figure 2: The normalized planetary and simulated asteroid data on a log-log scale.

Note that on the log-log scale Earth is located at $(0, 0)$ since the data is normalized relative to Earth. What is more interesting is that all the planets seem to be positioned along a straight line, and this is a strong evidence that the original data must satisfy a power law. It is also curious that there is a noticeable larger gap between Mars and Jupiter. This is the region of the solar system where the *asteroid belt* is located. It is believed that the asteroids in the main belt are remnants of a protoplanetary disk that never formed a planet, but they could also be the remnants of an ancient planet. The main belt, where the orbits are more stable, is between 2.2 and 3.3 AU from the Sun. For example, Ceres, which is the largest of the asteroids, about 1/13 the size of Earth, considered to be a dwarf planet, is located about 2.8 AU from the Sun, and has a period of 4.6 Earth years. Figure 2 shows a small sample from the asteroid belt between Mars and Jupiter, based on simulated data.

Exercise 3.2. Extend the R code from Exercise 3.1 to replicate Figure 2. To generate a random sample of average distances of the asteroids from the Sun, generate a sample of size 50 from the uniform distribution on the interval $(2.2, 3.3)$ with the help of the R function `runif()`, and use Kepler's third law to generate the corresponding periods. Use the `geom_point()` function to add the asteroids to the log-log plot of the planets. For the size of the points representing the asteroids, use the fact that the largest asteroids like Ceres are about 1/13 the size of Earth.

Note that Exercise 3.2 offers a fun way to exploit R's capabilities to create a visual representation of the asteroid belt. However, this is a departure from our efforts to "discover" Kepler's third law. Keep in mind that Figure 2 is not representative of the actual asteroids since we are not using empirical data, and we are actually using Kepler's third law to position them on the plot.

3.2 Kepler's 3rd Law from Normalized Planetary Data

As we already observed, Figure 2 shows that all planets are located very precisely along a straight line on the log-log scale, and this implies that the original data for distances and periods may satisfy a power law. Thus,

for the model function, we can use a power model with two parameters α and β:

$$T = \alpha D^{\beta}, \tag{1}$$

where T is the orbital period in Earth's years and D is the average distance from the Sun in AU. The data for Earth must satisfy this power model, but since the planetary data is normalized relative to Earth, both the Earth period $T_{\oplus} = 1$ and average distance from the Sun $D_{\oplus} = 1$, thus $\alpha = 1$, and we are left with only the β parameter to estimate from the normalized planetary data. We can linearize the power model (1) with $\alpha = 1$ by taking the natural logarithm on both sides. Now, we have an equation of a line on the log-log scale:

$$\ln(T) = \beta \ln(D) \tag{2}$$

This is a linear model with respect to the unknown parameter β. The response variable is now $y = \ln(T)$, and the predictor variable is $x = \ln(D)$. Thus, the linearized model can be written as:

$$y = \beta x \tag{3}$$

Instructional Note: Let us emphasize that if we do not normalize the planetary data, we would not be able to eliminate the parameter α the way we did. However, normalizing the data relative to Earth, prevents us from obtaining Kepler's law in its absolute form, being the more general power law (1). For instructional purpose, we consider both approaches, with and without normalization. The non-normalized case will be developed in Section 3.3.

Normalizing the data removes the parameter α, since if the original data satisfies (1), then the ratios, relative to Earth's data, satisfy the reduced model:

$$\frac{T}{T_{\oplus}} = \left(\frac{D}{D_{\oplus}}\right)^{\beta} \tag{4}$$

The reduced model (4) for the normalized data was linearized in (3), using a log-transformation on both variables. More generally, if N is the number of data points (9 in this case), how can we estimate the value of β from the values $x_j = \ln(D_j)$ and $y_j = \ln(T_j)$ for $j = 1, \ldots, N$?

Ideally, we want all pairs of x and y values to satisfy the model in (3), thus the linear system:

$$\begin{array}{rcl} \beta x_1 & = & y_1 \\ & \vdots & \\ \beta x_N & = & y_N \end{array} \iff \beta \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \iff \beta \mathbf{x} = \mathbf{y} \tag{5}$$

However, this overdetermined system has no solution in general. So, how do we "solve" a linear system that does not have a solution?

One geometrical way of thinking about finding a solution, in some sense, to a system like (5) that does not have a solution, is to use *orthogonal projections*. Observe that the linear system (5) does not have a solution, because the $\mathbf{y}$ vector in (5) is not a multiple of the $\mathbf{x}$ vector, in general. However, if we project orthogonally the $\mathbf{y}$ vector onto the $\mathbf{x}$ vector, as shown in Figure 3, then we can find a number β such that:

$$Proj_{\mathbf{x}}\mathbf{y} = \beta \mathbf{x} \tag{6}$$

The orthogonal projection of $\mathbf{y}$ onto $\mathbf{x}$ is done by taking the *dot product* of $\mathbf{y}$ with the unit vector $\frac{\mathbf{x}}{\|\mathbf{x}\|}$, and using the resulting number as a multiple for the unit vector $\frac{\mathbf{x}}{\|\mathbf{x}\|}$ to give us the projected vector:

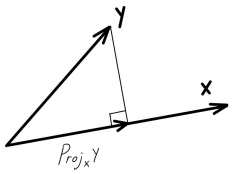

Figure 3: The orthogonal projection of $\mathbf{y}$ onto the line spanned by $\mathbf{x}$.

$$Proj_{\mathbf{x}}\mathbf{y} = \frac{\mathbf{x}\cdot\mathbf{y}}{\|\mathbf{x}\|}\frac{\mathbf{x}}{\|\mathbf{x}\|} = \frac{\mathbf{x}\cdot\mathbf{y}}{\|\mathbf{x}\|^2}\mathbf{x} = \frac{\mathbf{x}\cdot\mathbf{y}}{\mathbf{x}\cdot\mathbf{x}}\mathbf{x} \tag{7}$$

Compared with (6), the projection formula in (7) gives us at once a formula for the β value:

$$\beta = \frac{\mathbf{x}\cdot\mathbf{y}}{\mathbf{x}\cdot\mathbf{x}} = \frac{\sum_{j=1}^{N} x_j y_j}{\sum_{j=1}^{N} x_j^2} \tag{8}$$

Note that dividing by zero in (8) does not happen as long as we are dealing with a non-trivial problem having a non-zero vector $\mathbf{x}$. Also note that in the one-parameter linear model, the *line of best fit* $y = \beta x$ that we found, using an orthogonal projection, does not coincide with the line that goes through the center of mass of the data. It turns out that for linear models with two parameters, using orthogonal projections leads to the line of best fit, which does go through the center of mass of the data, as we show later.

There is an equivalent way of thinking about orthogonal projections in terms of minimizing the length of vectors. Consider an arbitrary multiple β of the given $\mathbf{x}$ vector, that is $\beta\mathbf{x}$, and take the difference with the given vector $\mathbf{y}$. Minimizing the Euclidean length of this difference $\|\mathbf{y} - \beta\mathbf{x}\|$ with respect to the parameter β is equivalent to finding the orthogonal projection of $\mathbf{y}$ onto $\mathbf{x}$ in which case the minimum length of the difference is achieved, i.e.

$$\text{Min}_{\beta}\,\|\mathbf{y} - \beta\mathbf{x}\| = \|y - Proj_{\mathbf{x}}\mathbf{y}\| \text{ for some } \hat{\beta} \text{ such that } Proj_{\mathbf{x}}\mathbf{y} = \hat{\beta}\mathbf{x} \tag{9}$$

We can generalize the idea of taking orthogonal projections to linear models with many parameters, which leads to multiple linear regression. Remember that a linear model in this context refers to being linear with respect to the model parameters.

We can compute orthogonal projections in R using the `project()` function from the `mosaic` R package. The `project()` function can be used for either:

1. Given `project(y,x)`, it projects `y` onto `x`, and returns the projected vector.
2. Given a formula (specified by `~`) as an argument, it works like the base R linear model function `lm()`, by constructing a model matrix whose columns are the vectors on the right-hand side of the formula for the more general case of multiple linear regression. For example, if we consider the formula `y~x1+x2+x3` for multiple linear regression, then `project(y~x1+x2+x3)` projects the vector `y` onto the column space of the model matrix whose columns are the vectors `x1`, `x2`, and `x3`. In this case, the result is the vector of the projection coefficients, for each of the column vectors in the model matrix.

In our case, we use option (2) with `project(y~x)` and since we are projecting the vector `y` onto the single vector `x`, the result is the single coefficient β such that $Proj_{\mathbf{x}}\mathbf{y} = \beta\mathbf{x}$:

```
x<-log(dataDT$D)    # x=log(D)
y<-log(dataDT$T)    # y=log(T)
beta<-project(y~x) # project y onto x to get the coefficient beta of x
```

This way, we get an estimate for $\beta \approx 1.4987$, and the fitted model becomes:

$$T = D^{1.4987} \tag{10}$$

Note that β is very close to 1.5, and the fitted model (10) is unitless since T and D are relative to Earth. This is a variation of what Kepler discovered about 400 years ago.

After laboring for 17 years on the observations of Brahe, Kepler discovered the Third Law of planetary motion on May 15, 1618. He wrote the following about this moment:

> "at first I believed I was dreaming ... But it is absolutely certain and exact that the proportion between the periodic times of any two planets is precisely the sesquialterate (in a ratio of three to two) proportion of their mean distances." From Kepler's *The Harmony of the World* (1619) [16].

However, Kepler does not reveal how he reached his conclusion that the power must be exactly 3/2.

3.3 Kepler's 3rd Law from Raw Planetary Data

Next, we explore the following question:

- What happens if we do not normalize the planetary data relative to Earth?

Now, we have to work with the raw `dist` and `period` data vectors, instead of the normalized vectors `D` and `T`. We also have to consider the more general power law (1) since we cannot imply anymore that $\alpha = 1$. We can linearize the power model (1), just like before, by taking the natural logarithm on both sides. We obtain again an equation of a line on the log-log scale, but this time with a non-zero y-intercept:

$$\ln(T) = \ln(\alpha) + \beta \ln(D) \tag{11}$$

This is still a linear model with respect to the unknown parameters $\gamma = \ln(\alpha)$ and β. The response variable is again $y = \ln(T)$, represented by the raw data vector `log(period)`, and the predictor variable is $x = \ln(D)$, represented by the raw data vector `log(dist)`. Thus, the linearized model can be written as:

$$y = \gamma + \beta x \tag{12}$$

Just as before, we want all pairs of x and y values to satisfy the model in (12), thus the linear system:

$$\begin{array}{rcl} \gamma + \beta x_1 & = & y_1 \\ & \vdots & \\ \gamma + \beta x_N & = & y_N \end{array} \iff \gamma \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} + \beta \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \iff \gamma \mathbf{1} + \beta \mathbf{x} = \mathbf{y} \tag{13}$$

Of course, this overdetermined system has no solution in general since the vector $\mathbf{y}$ is not a linear combination of the vectors $\mathbf{1}$ (the vector of ones) and $\mathbf{x}$, in general. So, again we want to "solve" a linear system that does not have a solution. We can generalize the idea of orthogonal projection, we used for the single parameter model. The idea is again to project orthogonally the vector $\mathbf{y}$, but this time onto the subspace $\mathrm{Im}(A) = \langle \mathbf{1}, \mathbf{x} \rangle$ spanned by the vectors $\mathbf{1}$ and $\mathbf{x}$, and let us denote this orthogonal projection by $\boldsymbol{\pi} = Proj_{\langle \mathbf{1}, \mathbf{x} \rangle} \mathbf{y}$, see Figure 4. In this notation, A represents the model matrix with columns $\mathbf{1}$ and $\mathbf{x}$. That way we get a consistent linear system that we can solve to find estimates for the two parameters. So, we want to solve the linear system:

$$\gamma \mathbf{1} + \beta \mathbf{x} = Proj_{\langle \mathbf{1}, \mathbf{x} \rangle} \mathbf{y} = \boldsymbol{\pi} \tag{14}$$

We assume that $\mathbf{1}$ and $\mathbf{x}$ are linearly independent, which is usually the case, provided that not all components of $\mathbf{x}$ are the same (having non-zero variance). Observe that γ and β are the coordinates of the projection relative to the vectors $\mathbf{1}$ and $\mathbf{x}$.

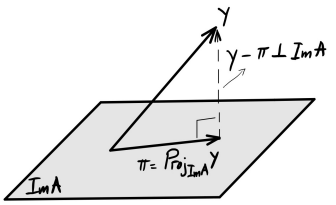

Figure 4: The orthogonal projection of $\mathbf{y}$ onto $\mathrm{Im}(A) = \langle \mathbf{1}, \mathbf{x} \rangle$.

Assume now that the components of $\mathbf{x}$ sum to zero, that is, $\sum_{j=1}^{N} x_j = 0$, which is equivalent to $\bar{x} = 0$, or $\mathbf{1} \cdot \mathbf{x} = 0$. Since the dot product is zero, the vectors are orthogonal. Thus, the orthogonal projection of $\mathbf{y}$ onto the subspace $\langle \mathbf{1}, \mathbf{x} \rangle$ can be found by computing the individual orthogonal projections onto the basis vectors.

Exercise 3.3. Take the dot product of (14) first with $\mathbf{1}$ and then with $\mathbf{x}$, and use that $\mathbf{1} \cdot \mathbf{x} = 0$ to derive:

$$\gamma = \frac{\boldsymbol{\pi} \cdot \mathbf{1}}{\mathbf{1} \cdot \mathbf{1}}, \quad \beta = \frac{\boldsymbol{\pi} \cdot \mathbf{x}}{\mathbf{x} \cdot \mathbf{x}} \tag{15}$$

Note that $\mathbf{1} \cdot \mathbf{1} = N$ and $\mathbf{x} \cdot \mathbf{x} = \|\mathbf{x}\|^2$.

Exercise 3.4. Since $\boldsymbol{\pi}$ is the orthogonal projection of $\mathbf{y}$ on the subspace $\langle \mathbf{1}, \mathbf{x} \rangle$, it follows that $\mathbf{y} - \boldsymbol{\pi} \perp \langle \mathbf{1}, \mathbf{x} \rangle$. Thus, $\mathbf{y} - \boldsymbol{\pi} \perp \mathbf{1}$ and $\mathbf{y} - \boldsymbol{\pi} \perp \mathbf{x}$, but this implies $(\mathbf{y} - \boldsymbol{\pi}) \cdot \mathbf{1} = 0$ and $(\mathbf{y} - \boldsymbol{\pi}) \cdot \mathbf{x} = 0$. Use this observation and (15) to get the following formulas for the parameters:

$$\gamma = \frac{\mathbf{y} \cdot \mathbf{1}}{N} = \bar{y}, \quad \beta = \frac{\mathbf{y} \cdot \mathbf{x}}{\|\mathbf{x}\|^2} \tag{16}$$

Thus, the line of best fit to the data when $\bar{x} = 0$, based on orthogonal projections, is given by:

$$y = \bar{y} + \frac{\mathbf{y} \cdot \mathbf{x}}{\|\mathbf{x}\|^2} x, \text{ provided that } \bar{x} = 0 \tag{17}$$

In the general case when $\bar{x} \neq 0$, we can simply shift the x variable by the sample mean $\bar{x}$ of the data vector $\mathbf{x}$. More generally, we consider the linear (w.r.t γ and β) model:

$$y = \gamma + \beta(x - \bar{x}) \tag{18}$$

Exercise 3.5. Use the projection $\boldsymbol{\pi} = Proj_{\langle \mathbf{1}, \mathbf{x} - \bar{x}\mathbf{1} \rangle} \mathbf{y} = \gamma \mathbf{1} + \beta(\mathbf{x} - \bar{x}\mathbf{1})$, where $\mathbf{x} - \bar{x}\mathbf{1}$ is the vector of x coordinates shifted by the sample mean $\bar{x}$, and follow the logic and the methods we used when $\bar{x} = 0$ to show the following:

1. $\mathbf{1} \perp (\mathbf{x} - \bar{x}\mathbf{1})$ (19)
2. $(\mathbf{y} - \boldsymbol{\pi}) \perp \mathbf{1}$ and $(\mathbf{y} - \boldsymbol{\pi}) \perp (\mathbf{x} - \bar{x}\mathbf{1})$ (20)
3. $\gamma = \dfrac{\mathbf{y} \cdot \mathbf{1}}{N} = \bar{y}, \quad \beta = \dfrac{\mathbf{y} \cdot (\mathbf{x} - \bar{x}\mathbf{1})}{(\mathbf{x} - \bar{x}\mathbf{1}) \cdot (\mathbf{x} - \bar{x}\mathbf{1})} = \dfrac{\mathbf{y} \cdot \mathbf{x} - N\bar{x}\bar{y}}{\|\mathbf{x}\|^2 - N(\bar{x})^2} = r_{xy} \dfrac{s_y}{s_x}$ (21)
4. the line of best fit (the regression line): $\boxed{y = \bar{y} + r_{xy} \dfrac{s_y}{s_x}(x - \bar{x})}$, (22)

where r_{xy} is the sample correlation between the data vectors $\mathbf{x}$ and $\mathbf{y}$; s_y and s_x are the sample standard deviations whose definitions are given below.

$$r_{xy} = \frac{(\mathbf{x} - \bar{x}\mathbf{1}) \cdot (\mathbf{y} - \bar{y}\mathbf{1})}{\|\mathbf{x} - \bar{x}\mathbf{1}\| \, \|\mathbf{y} - \bar{y}\mathbf{1}\|} = \frac{\mathbf{y} \cdot \mathbf{x} - N\bar{x}\bar{y}}{(N-1)s_x s_y} \tag{23}$$

$$s_x^2 = \frac{1}{N-1}\|\mathbf{x} - \bar{x}\mathbf{1}\|^2 = \frac{1}{N-1}(\|\mathbf{x}\|^2 - N(\bar{x})^2) \tag{24}$$

Note that the line of best fit in (22) implies that the center of mass of the data $(\bar{x}, \bar{y})$ lies on the line. Also, the sample correlation has a simple geometrical interpretation, namely: it is the orthogonal projection of the centralized (mean zero) vector $\mathbf{y} - \bar{y}\mathbf{1}$ onto the centralized (mean zero) vector $\mathbf{x} - \bar{x}\mathbf{1}$, both normalized to unit vectors. This makes it clear why the sample correlation is in the range $[-1, 1]$. The sample standard deviation is simply the normalized length of the centralized, mean zero, vector.

We can now fit the model in (18) to the raw planetary data, either by using the formulas from Exercise 3.5 or by using the R function `project()` from the `mosaic` package.

```
y<-log(period) # log of raw period data
x<-log(dist)   # log of raw distance data
xc<-x-mean(x) # centralized, mean zero vector
project(y~1+xc) # project y onto the span of 1 and xc
```

```
## (Intercept)          xc
##    8.080194    1.498800
```

In the function call `project(y~1+xc)`, the vector `xc` has been centralized to have mean zero, and the formula object `y~1+xc` means to project the vector `y` onto the span of the vectors `1` and `xc`. Note that R turns the number `1` into a vector of ones, of the same size as `xc`. The function call returns the coordinates γ and β, w.r.t. the basis vectors `1` and `xc`, being the y-intercept and the slope parameters of the linear model in (18).

The results from Exercise 3.5 give us another way to compute the parameters, as shown below.

```
gamma <- mean(y) # y-intercept = sample mean of y
beta <- cor(x,y)*sd(y)/sd(x) # slope of line of best fit
```

In the code chunk above, we used the base R function `cor(x,y)`, which computes the sample correlation between `x` and `y`, as well as the `sd(y)`, which computes the sample standard deviation of `y`.

Using the optimal values for $\gamma = \bar{y} = 8.0802$ and $\beta = r_{xy}\frac{s_y}{s_x} = 1.4988$, we can exponentiate the fitted linear model (18). In this notation, T and D represent the original (not normalized) period and distance variables.

$$\ln(T) = \gamma + \beta(\ln(D) - \bar{x}) \implies T = e^{\bar{y} - \beta\bar{x}} D^\beta = \alpha D^\beta, \tag{25}$$

where $\beta \approx 1.4988$, and $\alpha = e^{\bar{y} - r_{xy}\frac{s_y}{s_x}\bar{x}} \approx 0.2007$.

In (25), T, α and D have units, and one can use *dimensional analysis* to show that β must be exactly $3/2$. Dimensional analysis can be very useful to check the correctness of an equation that we have derived after some algebraic manipulations. More importantly, dimensional analysis can be used to determine the form of an equation itself. This goes a bit beyond the scope of our presentation as it requires some additional physics knowledge, but we shall try to illustrate its use in the context of Kepler's 3rd Law. The key observation is that most physical quantities can be expressed in terms of combinations of basic dimensions such as mass (M), length (L), time (T), etc. Note that the term "dimension" is not quite the same as a "unit", but they are closely related. For example, the unit of force is Newton (N), which can be expressed in terms of more basic units as $\mathrm{N} = \mathrm{kg} \times \mathrm{m} \times \mathrm{s}^{-2}$, and the dimension of force is MLT^{-2}. The key step in dimensional analysis is to guess the form of a given quantity of interest as a power law involving all quantities that the main quantity should depend on. The period P of a planet orbiting the Sun should depend on the average distance D to the center of the Sun, the mass M_0 of the Sun, and the gravitational constant G. Note that we used

the letter P to represent the orbital period, rather than T, since we want to use T to represent the time dimension in our dimensional analysis. The most general power law that involves these quantities is given by:

$$P = kM_0^a D^b G^c, \tag{26}$$

where k is a dimensionless constant, and a, b and c are powers to be determined. The dimensions of G are $[\mathrm{N}]L^2M^{-2}$, which can be determined from Newton's Law of Gravity, where the dimensions of force are $[\mathrm{N}] = MLT^{-2}$, so the dimensions of G are $MLT^{-2}L^2M^{-2} = L^3M^{-1}T^{-2}$. Substitute in (26) to get:

$$T = kM^a L^b (L^3 M^{-1} T^{-2})^c = kM^{a-c} L^{b+3c} T^{-2c}$$

Comparing terms and powers yields:

$$a - c = 0,\ b + 3c = 0,\ 1 = -2c \implies a = c = -1/2,\ b = 3/2$$

Thus, the power law in (26) becomes:

$$P = \frac{k}{\sqrt{M_0 G}} D^{3/2}, \tag{27}$$

And Kepler's 3rd law of planetary motion emerges from (27):

$$D^3 = \frac{GM_0}{k^2} P^2 \tag{28}$$

If we consider the special case of a circular motion around the Sun, we can get that $k^2 = 4\pi^2$. Note that this dimensional analysis requires knowing Newton's Law of Gravity.

Exercise 3.6. Create a plot of the raw planetary data and superimpose the graph of the fitted model (25), as shown in Figure 5. Optionally, add some random asteroids that obey Kepler's 3rd law.

Solution. The code for Exercise 3.6 is given below.

```
## Solution to Exercise 3.6
# colors for the planets
colors<-c("black","brown","blue","red","orange","gold","lightblue","darkblue","black")
raw_data<-tibble(dist,period) # raw planetary data
y<-log(period)  # log of raw period data
x<-log(dist)  # log of raw distance data
beta <- cor(x,y)*sd(y)/sd(x)  # slope of line of best fit
a<-exp(mean(y) - cor(x,y)*sd(y)/sd(x)*mean(x))
K<-1/a^2  # Kepler's constant without converting 10^6 km to AU
# D for model graph in 10^6 km, relative to Earth
Dmodel <- dist[3]*seq(from=0.1, to=40, length = 200)
Tmodel <- a*Dmodel^beta  # T for model graph in days
grid <- tibble(Dmodel,Tmodel)  # grid of D and T values for the model
# Optionally add some random asteroids using Kepler's 3rd law
N<-30  # number of random asteroids to plot
Dasteroids <-dist[3]*runif(N,2.2,3.3)  # random distances in 10^6 km
Tasteroids <- a*Dasteroids^beta  # Kepler's 3rd law for the asteroid periods
f <- 1.4  # a multiple to scale the planet sizes
sizes<-f*c(1/3,0.9,1,1/2,11,9,4,3.9,1/5)  # planet sizes
# Create the plot
ggplot() +
```

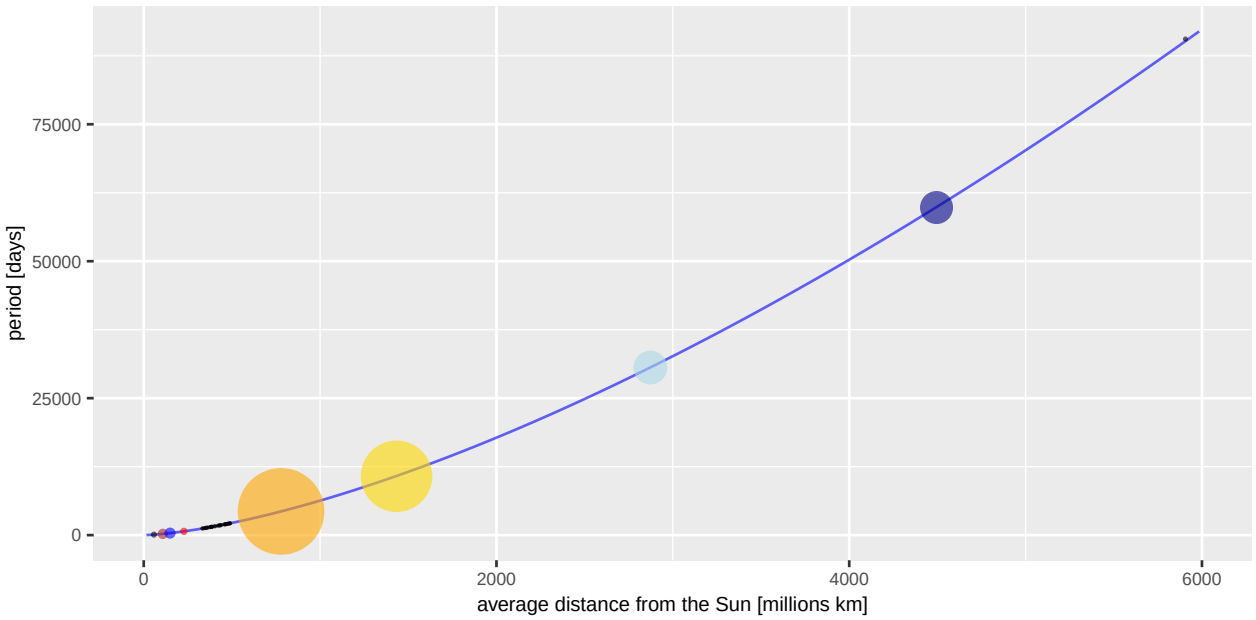

Figure 5: The fitted power model superimposed on the raw planetary data.

```
geom_line(mapping=aes(x=Dmodel,y=Tmodel),data=grid,col="blue",size=0.5,alpha=0.6)+
geom_point(mapping=aes(x=dist,y=period),data=raw_data,col=colors,size=sizes,alpha=0.6)+
labs(x ="average distance from the Sun [millions km]",y = "period [days]")+
geom_point(mapping=aes(x=Dasteroids,y=Tasteroids),col="black",size=1/13*f,alpha=0.6)+
theme(axis.text=element_text(size=7), axis.title=element_text(size=8))
```

4 Discovering Kepler's 1st Law

4.1 Kepler's Measurements of Mars

Project Mosaic [15] has on its website some legacy datasets that are not part of the `mosaicData` package, which gets installed along with the `mosaic` package. One of the datasets, named `kepler-mars.csv` contains measurements for the polar coordinates of Mars relative to the Sun, on the orbital plane of Mars. In addition to the historical measurements, the data also contain a modern reconstruction of the "actual" positions of Mars at the time Tycho Brahe made his very precise measurements, before even the telescope was invented.

In Figure 6, Mars is shown only once as the point M, while the Earth's position is shown as E_1 and E_2, being the positions of Earth at the start and end of one Mars year, which Kepler estimated to be 687 Earth days. The angles $\angle ME_1S$ and $\angle ME_2S$ can be determined by observation, as can $\angle E_1SE_2$ from knowledge of one Mars year. Trigonometry can then be used to solve for the Mars' distance r from the Sun. When Kepler used this triangulation for Mars, he discovered that its distance from the Sun varies, and that its orbit is an ellipse with the Sun at one of its foci. In fact, it was Kepler who named the two centers in the ellipse "foci" ("focus" in Latin means "fireplace").

Exercise 4.1. Read [9] to understand how Kepler did his triangulation of Mars.

Exercise 4.2. Load the `kepler-mars.csv` dataset from the `Data` tab on the Project Mosaic website [15], and create the dataframe `Kepler` in RStudio, using the function `read.csv()`, having as argument the web address of the dataset.

Solution. Run the code below to load the `kepler-mars.csv` dataset and create the `Kepler` dataframe.

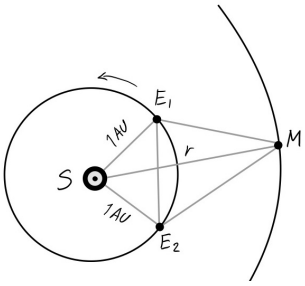

Figure 6: Planetary triangulation used by Kepler to find the distance of Mars from the Sun.

```
Kepler<-read.csv("http://www.mosaic-web.org/go/datasets/kepler-mars.csv")
```

The dataframe `Kepler` has 28 observations and 6 variables, but we consider only the following three variables:

- `kepler.radius`: The radius (r) of Mars, measured from the Sun in AU.
- `kepler.angle`: The polar coordinates angle (θ) on the orbital plane of Mars, measured in radians, known as "true anomaly".
- `time`: The time interval in Earth days for the measurements, relative to a fixed day (presumably from the time Brahe made the original measurements).

The computational explorations in the next two sections were inspired by the *MOSAIC Calculus* text [11], which we used for a Calculus Lab.

4.2 Fitting an Ellipse to Kepler's Mars Data

Consider an ellipse with the Sun being in one of its foci. The polar form of the ellipse relative to the focus where the Sun is located, as the center of the coordinate system, is given in terms of the radius r (originating from that focus of the ellipse) as a function of the angle θ, measured relative to the major axis, which is related to the so called true anomaly.

$$r = \frac{a}{1 + e\cos(\theta)} \tag{29}$$

This form assumes that the reference direction $\theta = 0$ points towards the center. The parameters a and e describe the shape of the ellipse, and e is called the **eccentricity**. Having zero eccentricity turns the ellipse into a circle of radius a. The equation of the ellipse in polar form given in (29), corresponds to a coordinate system centered at one of the foci, with the angle θ measured, as shown in Figure 7.

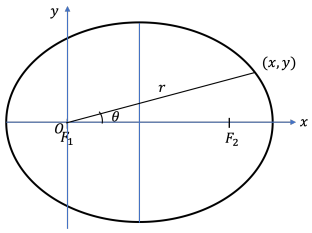

Figure 7: An ellipse in polar form with the Sun in the left focus.

Exercise 4.3. Derive the polar form of the ellipse given in (29). As a reference, use Section 9.4 *Conics in Polar Coordinates* in [20].

Exercise 4.4. Create a scatterplot of Mars' radius from the Sun, given by the variable `kepler.radius`, against the corresponding polar angle, given by the variable `kepler.angle`, as shown in Figure 8. Use the `gf_point()` plotting function from the `ggformula` package that gets installed with the `mosaic` package.

Solution. The code below creates the scatterplot in Figure 8. Note that the function `gf_theme()` is a helper function used to control the size of the axis titles and axis text (marks), and `|>` is the pipe operator, which implements composition of functions.

```
gf_point(kepler.radius ~ kepler.angle, col="red", size=0.4, data=Kepler,
         xlab="Kepler's angle [radians]", ylab="Kepler's radius [AU]") |>
  gf_theme(axis.title = element_text(size = 7), axis.text = element_text(size = 6))
```

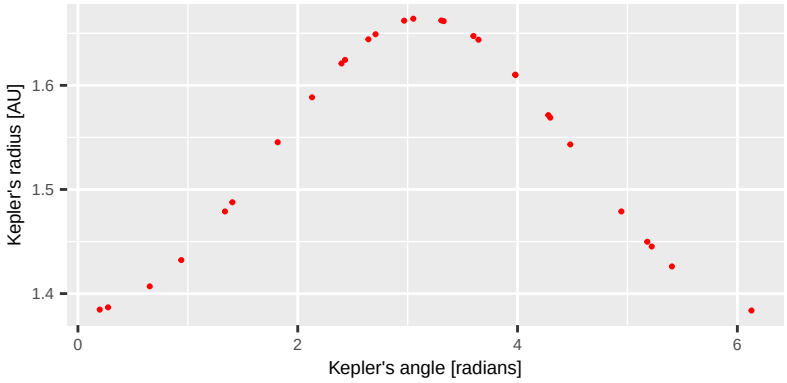

Figure 8: A scatterplot of Kepler's radius r, againsts Kepler's angle θ.

Exercise 4.5. Estimate the values of the parameters a and e by fitting the model in (29) to Kepler's data, using nonlinear least squares since the model depends nonlinearly on e (it depends linearly on a). Use the `fitModel()` function from the `mosaic` package, and the fact that `kepler.radius` corresponds to the variable r, and `kepler.angle` corresponds to the variable θ. The fitted model has the form:

$$r = \frac{1.5104}{1 + 0.0926\cos(\theta)} \tag{30}$$

Solution. Using the `fitModel()` function to fit the model in (29) to Kepler's data is implemented by the code below, where the formula `kepler.radius ~ a/(1 + e*cos(kepler.angle))` specifies the model with the two parameters `a` and `e`. We provide starting values for `a` and `e` using the `start` argument, where we take the average Kepler's radius as the initial value of `a`, and zero as the initial value of `e`, given that the orbit of Mars is close to a circle. The resulting object `r_fun` is an R function with argument θ, i.e. $r(\theta)$. The vector `params` contains the fitted parameters. In physics texts, the eccentricity of Mars is given as 0.093.

```
# fit an ellipse in polar coordinates to Kepler's data
r_fun <- fitModel(kepler.radius ~ a/(1 + e*cos(kepler.angle)),
                  start=list(a=mean(Kepler$kepler.radius),e=0), data=Kepler)
params<-coef(r_fun) # fitted parameters
params # print fitted coefficients
```

```
##          a          e
## 1.51043091 0.09263881
```

Consider a Cartesian coordinate system where the Sun is located at the center $(0, 0)$, and the position of Mars going around the Sun is given in polar coordinates by (r, θ). We can transform the polar (r, θ) into Cartesian (x, y) coordinates using the relations $x = r\cos(\theta)$ and $y = r\sin(\theta)$.

Exercise 4.6. Transform the polar coordinates of Mars `kepler.radius` and `kepler.angle` into Cartesian coordinates. Add the vectors of x and y coordinates to the `Kepler` dataframe, using the `mutate()` function

from the `dplyr` package in the `tidyverse` collection.

Solution. The transformation from polar to Cartesian coordinates is implemented by the code below, where `mutate` allows us to add to the `Kepler` dataframe two new variables, `x` and `y`, which contain the x and y coordinates obtained from the corresponding polar coordinates.

```
Kepler <- Kepler |>
            mutate(x=kepler.radius*cos(kepler.angle),y=kepler.radius*sin(kepler.angle))
```

Exercise 4.7. The goal of this exercise is to plot the graph of the fitted ellipse (30) on top of Kepler's data:

- Plot the positions of Mars with `gf_point()`, using their x and y coordinates from `Kepler`.
- Add the Sun at the origin $(0, 0)$ using the formula `0~0` with `col="yellow"` and big size (`size=12`).
- Use the R function `r_fun()` for $r(\theta)$ returned by `fitModel()` to generate the vectors of x and y coordinates for a vector of θ values, generated with `theta<-seq(0,2*pi,len=200)` in radians. Use `gf_point()` to plot these 200 points that represent the fitted orbit of Mars, and replicate Figure 9.

Solution. Figure 9 is created with the code below. Note that the object `r_fun`, returned by the function `fitModel()`, is actually an R function $r(\theta)$ that implements the fitted ellipse in (30). In particular, we can compute the Cartesian coordinates of the orbit of Mars using `r_fun(theta)*cos(theta)` for the x coordinates, and `r_fun(theta)*sin(theta)` for the y coordinates, where `theta` is a vector of 200 numbers from $[0, 2\pi]$. Note that `0~0` plots a single point at $(0, 0)$.

```
# plot Kepler's data, the Sun, and the orbit of Mars
theta<-seq(0,2*pi,length=200) # 200 angle values
orbit<-tibble(xOrbit=r_fun(theta)*cos(theta),yOrbit=r_fun(theta)*sin(theta))
gf_point(yOrbit ~ xOrbit, data=orbit, col="pink", size=0.4, alpha=0.5) |>
  gf_point(y ~ x, data=Kepler, col="red", size=0.8) |>
  gf_point(0 ~ 0, size=12, col="yellow") +
  theme_void()
```

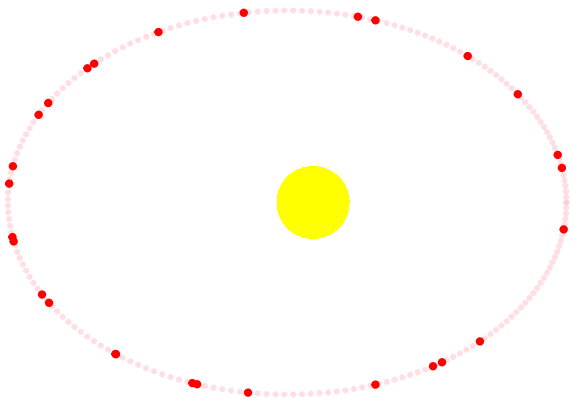

Figure 9: The fitted orbit of Mars superimposed on Kepler's data.

5 Discovering Kepler's 2nd Law

In this section, we want to explore how the area dS swept by the radius vector joining Mars to the Sun depends on the small time interval dt. For this purpose, we want to use the 28 data points of `kepler.angle` and `time`, and create a continuous function of time using linear interpolation of our sparse data. We can create an R function `theta_fun()`, which interpolates the data linearly, by using the `connector()` function from the `mosaic` package, shown in the code below.

```
theta_fun <- connector(kepler.angle ~ time, data=Kepler)
```

In Figure 10, we visualize the graph of the function `theta_fun` using the `slipe_plot()` function from `mosaicCalc`, and superimpose it over the data points. Note where the data points are located.

```
gf_point(kepler.angle ~ time, data=Kepler, size=0.6, col="blue") |>
slice_plot(theta_fun(t) ~ t, domain(t=c(-789,4000)),
           size=0.6, col="blue", alpha=0.5, n=1000) |>
  gf_labs(x = "time [days]", y="theta [radians]") +
  theme(axis.text=element_text(size=7), axis.title=element_text(size=8))
```

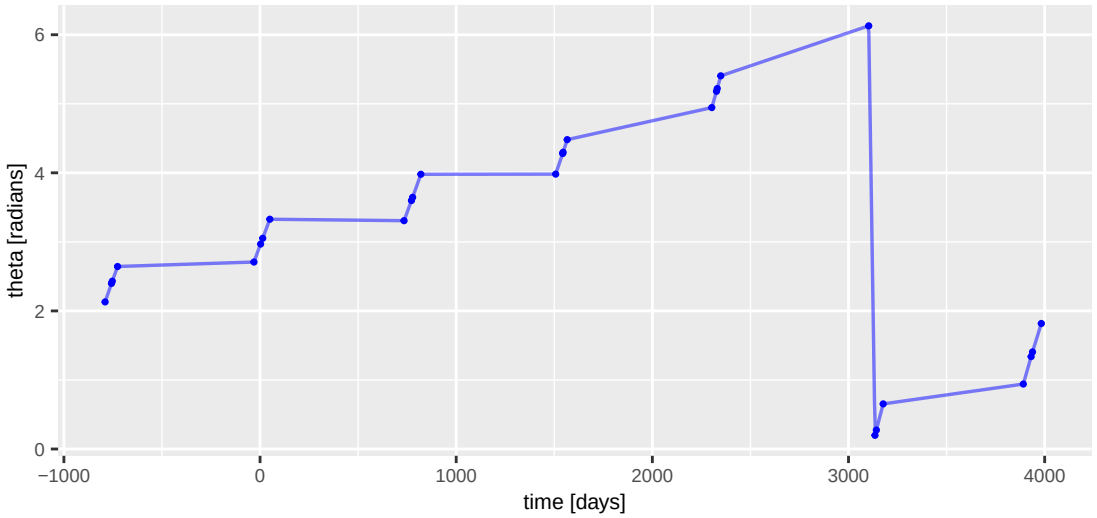

Figure 10: The graph of $\theta(t)$ superimposed over the data points.

We use the R function `r_fun()`, returned by `fitModel()`, which implements the fitted ellipse $r(\theta)$ in (30) to construct functions $x(t)$ and $y(t)$ that give the x and y coordinates of Mars' orbital position at time t.

$$x(t) = r(\theta(t))\cos(\theta(t)) \tag{31}$$
$$y(t) = r(\theta(t))\sin(\theta(t)) \tag{32}$$

The function $\theta(t)$ is represented by the R function `theta_fun`, and $r(\theta(t))$ can be implemented by taking the composition of `r_fun` and `theta_fun`. The R functions `x_fun` and `y_fun` that implement $x(t)$ and $y(t)$ are given by the code below, where we use the `makeFun()` function from the `mosaic` package.

```
x_fun <- makeFun(r_fun(theta_fun(t))*cos(theta_fun(t)) ~ t)
y_fun <- makeFun(r_fun(theta_fun(t))*sin(theta_fun(t)) ~ t)
```

Now, we have the ability to generate the x and y coordinates of the point on the fitted ellipse at time t as well as time $t + dt$ for some small time interval dt. This allows us to compute the area of the small triangle formed by the Sun and Mars at the two positions along the orbit. If the time interval dt is small enough then the area of the triangle will be approximately equal to the area of the segment cut from the ellipse by the radius vector connecting the Sun with Mars. The area of the triangle is given by:

$$dS = \frac{1}{2}\det\begin{vmatrix} x_i & x_{i+1} \\ y_i & y_{i+1} \end{vmatrix} = \frac{1}{2}(x_i y_{i+1} - x_{i+1} y_i), \tag{33}$$

where (x_i, y_i) and (x_{i+1}, y_{i+1}) are the coordinates of the points on the ellipse at times t and $t + dt$.

Next, we sample 5000 random time points t within the range of the variable `time`, using the random number generator of the Uniform distribution on the interval $[-790, 3000]$. Similarly, we generate 5000 random values

for the small time interval dt, sampled from the Uniform distribution on the interval $[0.1, 0.5]$ to represent a small enough time interval between 1/10 and 1/2 of a day. Then, we can find the area dS for all 5000 pairs of t and dt by using the `x_fun()` and `y_fun()` functions evaluated at t and $t + dt$.

```
set.rseed(2022) # set the random seed
t <- runif(5e3,-790,3000) # random times
dt <- runif(5e3,0.1,0.5) # random small time intervals
dS<-0.5*(x_fun(t)*y_fun(t+dt)-x_fun(t+dt)*y_fun(t)) # area of segment
```

Now, we can visualize the relationship between the area of the segment dS and the small time interval dt by plotting `dS` against `dt`, shown in Figure 11.

```
mydata <-tibble(dS,dt) # bind dS and dt into a dataframe
gf_point(dS ~ dt, data=mydata, pch=20, size=0.1, col="blue", alpha=0.7) |>
  gf_labs(x = "dt [days]", y="dS") +
  theme(axis.text=element_text(size=7), axis.title=element_text(size=8))
```

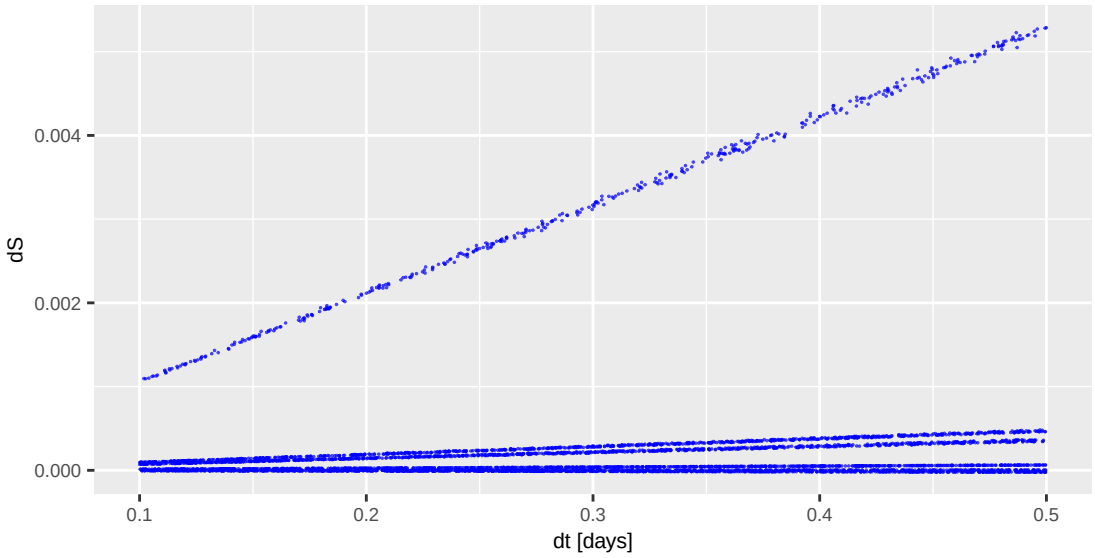

Figure 11: The scatterplot of dS against dt for a random sample of size 5000.

Notice how the sample of size 5000 gets localized into 6 fairly well-defined linear trends. The bottom 5 trend-lines with the smallest slopes appear to correspond to the parts from the graph of $\theta(t)$ in Figure 10 within day 3000 that are either almost flat or have a small positive slope. Note that the bottom two trendlines over the x-axis can be distinguished on a bigger plot or if you zoom into the plot along the y-axis. The top-most linear trend with the highest slope corresponds to the parts of the graph of $\theta(t)$ where the actual data points are located, as can be seen from Figure 10, and this is the linear trend that we want to keep since the other 5 linear trends appear to be artifacts of the linear interpolation of our sparse data. We can filter the data for `dS` to extract only the top linear trend by observing that all values of `dS` for this linear trend appear to be above 0.001. We can fit a line to the filtered data to find the slope of the linear trend.

```
# filter the data
mydata <- mydata |>
  filter(dS > 0.001)
# linear regression on the filtered data
model<-lm(dS ~ dt, data=mydata)
```

We can add the regression line to the scatterplot of the filtered data with the code below.

```
gf_point(dS ~ dt, data=mydata, pch=20, size=0.1, col="blue", alpha=0.8) |>
  gf_lm(col="red", lwd=0.3, alpha=0.8) |>
  gf_labs(x = "dt [days]", y="dS") +
  theme(axis.text=element_text(size=7), axis.title=element_text(size=8))
```

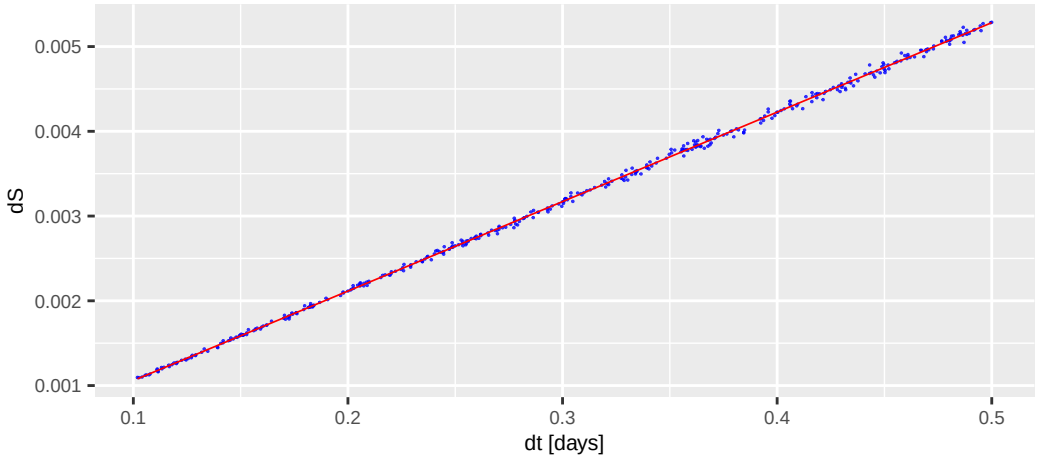

Figure 12: The regression line with slope $k_0 = 0.01056$, fitted to the filtered sample.

The linear regression fit gives a slope of $k_0 = 0.01056$, which can be accessed with `coef(model)`.

Figure 12 implies that the rate of change (with respect to time) of the area swept by the radius vector of the planet's orbit is approximately equal to the constant k being the slope of the trendline, i.e. $\frac{dS}{dt} \approx k_0$.

The exact statement of Kepler's 2nd law in this context can be stated as follows:

$$\frac{dS}{dt} = \text{const} = k \implies \int_0^T dS = \int_0^T k dt \implies \pi AB = kT, \tag{34}$$

where T is the orbital period of Mars, πAB is the area of the ellipse, with A being the semimajor axis, and B the semiminor axis. Solving for k gives:

$$k = \frac{\pi AB}{T} = 0.01058, \tag{35}$$

The numerical value of k in (35) is obtained using the official values for A, B and T given below.

```
A <- 1.52400 # semimajor axis of Mars in AU
B <- 1.51740 # semiminor axis of Mars in AU
T <- 686.980 # orbital period of Mars in Earth days
```

Note that the value k_0 obtained from the linear regression is remarkably close to the theoretical value of k obtained from Kepler's second law.

6 Conclusions

Fitting models to data is fundamentally important to modern data science, and we hope that these activities designed to be used at lower-level undergraduate courses, can help students build mathematical intuition and computing skills, before they tackle more advanced topics in data science.

For modern data science applications, using big, complex data, we invite the reader to explore further the `tidyverse` [26] collection of packages, developed by RStudio, for doing data analysis, visualizations, modeling

and machine learning, using tidyverse principles. For a deeper dive into modern data science, we encourage the reader to explore the freely available online books, *Data Science: A First Introduction* [24], *Modern Data Science with R* [3], and *R for Data Science* [27].

References

[1] JJ Allaire. *RStudio*. 2021. URL: https://www.rstudio.com/products/rstudio/.

[2] JJ Allaire. *RStudio Cloud*. 2022. URL: https://rstudio.cloud/.

[3] Benjamin S. Baumer, Daniel T. Kaplan, and Nicholas J. Horton. *Modern Data Science with R*. Second. CRC Press, 2021.

[4] Nadia Benakli et al. "Introducing computational thinking through hands-on projects using R with applications to calculus, probability and data analysis". In: *International Journal of Mathematical Education in Science and Technology* 48.3 (2017). Publisher: Taylor & Francis, pp. 393–427. DOI: 10.1080/0020739X.2016.1254296. URL: https://doi.org/10.1080/0020739X.2016.1254296.

[5] W. John Braun and Duncan J. Murdoch. *A First Course in Statistical Programming with R*. en. Cambridge University Press, 2021. DOI: 10.1017/9781108993456.

[6] Britannica. *Kepler's Laws of Planetary Motion*. URL: https://www.britannica.com/science/Keplers-laws-of-planetary-motion.

[7] John Chambers. *The R Project for Statistical Computing*. 2022. URL: https://www.r-project.org/.

[8] Jonathan Cornelissen. *Introduction to R Online - DataCamp Learn*. Web course. 2022. URL: https://app.datacamp.com/learn/courses/free-introduction-to-r.

[9] Owen Gingerich. "The Great Martian Catastrophe and How Kepler Fixed It". In: *Physics Today* 64.9 (2011), pp. 50–54. DOI: 10.1063/PT.3.1259.

[10] Garrett Grolemund, Xie Yihui, and J. J. Allaire. *R Markdown: The Definitive Guide*. CRC Press, 2021. URL: https://bookdown.org/yihui/rmarkdown/.

[11] Daniel Kaplan. *MOSAIC Calculus*. 2022. URL: http://www.mosaic-web.org/MOSAIC-Calculus/index.html (visited on 09/27/2022).

[12] Daniel Kaplan, Cecylia Bocovich, and Randall Pruim. "Modeling-based Calculus with R/mosaic." In: *The UMAP Journal* 36.1 (2015), p. 29. URL: https://www.comap.com/product/?idx=1470.

[13] Daniel Kaplan and Randall Pruim. *ggformula: Formula Interface to the Grammar of Graphics*. 2021. URL: https://CRAN.R-project.org/package=ggformula.

[14] Daniel T. Kaplan, Randall Pruim, and Nicholas J. Horton. *mosaicCalc: Function-Based Numerical and Symbolic Differentiation and Antidifferentiation*. 2020. URL: https://CRAN.R-project.org/package=mosaicCalc.

[15] Daniel T. Kaplan, Randall Pruim, and Nicholas J. Horton. *Project MOSAIC*. URL: http://www.mosaic-web.org/ (visited on 09/27/2022).

[16] Johannes Kepler. *The Harmony of the World*. American Philosophical Society, 1997.

[17] Boyan Kostadinov. *R Markdown notebook for this article*. URL: https://github.com/bkostadi/Kepler.git.

[18] Boyan Kostadinov, Johann Thiel, and Satyanand Singh. "Creating Dynamic Documents with R and Python as a Computational and Visualization Tool for Teaching Differential Equations". In: *PRIMUS* 29.6 (2019). Publisher: Taylor & Francis, pp. 584–605. DOI: 10.1080/10511970.2018.1472161.

[19] Samuel Ling. *Physics Bootcamp: Kepler's Laws of Planetary Motion*. 2022. URL: http://www.physicsbootcamp.org/Keplers-Laws-of-Planetary-Motion.html (visited on 09/27/2022).

[20] David Lippman and Melonie Rasmussen. *Precalculus*. Second. Tangential Projects Press, 2022.

[21] Randall Pruim, Daniel T. Kaplan, and Nicholas J. Horton. "The mosaic Package: Helping Students to Think with Data Using R". en. In: *The R Journal* 9.1 (2017), pp. 77–102. URL: https://journal.r-project.org/archive/2017/RJ-2017-024/index.html.

[22] RStudio. *R Markdown*. URL: https://rmarkdown.rstudio.com/lesson-1.html (visited on 09/27/2022).

[23] Frank Swetz. "Johannes Kepler's Astronomia Nova". In: *Convergence* (2010). DOI: 10.4169/loci003443. URL: https://www.maa.org/publications/periodicals/convergence/johannes-keplers-astronomia-nova (visited on 09/27/2022).

[24] Tiffany Timbers, Trevor Campbell, and Melissa Lee. *Data Science: A First Introduction*. New York: Chapman and Hall/CRC, 2022. DOI: 10.1201/9781003080978.

[25] Kevin Ushey, JJ Allaire, and Yuan Tang. *Reticulate: an R Interface to Python*. URL: https://rstudio.github.io/reticulate/.
[26] Hadley Wickham. *Tidyverse: Easily Install and Load the Tidyverse*. 2021.
[27] Hadley Wickham and Garrett Grolemund. *R for Data Science*. O'Reilly Media, 2017. URL: https://r4ds.had.co.nz/.
[28] Hadley Wickham et al. *dplyr: A Grammar of Data Manipulation*. 2021. URL: https://CRAN.R-project.org/package=dplyr.
[29] Hadley Wickham et al. *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*. 2021. URL: https://CRAN.R-project.org/package=ggplot2.
[30] Hadley Wickham et al. "Welcome to the Tidyverse". In: *Journal of Open Source Software* 4.43 (2019), p. 1686. DOI: 10.21105/joss.01686.
[31] David Williams. *Earth Fact Sheet*. 2022. URL: https://nssdc.gsfc.nasa.gov/planetary/factsheet/earthfact.html (visited on 09/27/2022).
[32] David Williams. *Planetary Fact Sheet*. 2022. URL: https://nssdc.gsfc.nasa.gov/planetary/factsheet/ (visited on 09/27/2022).
[33] Yihui Xie, Christophe Dervieux, and Emily Riederer. *R Markdown Cookbook*. CRC Press, 2022. URL: https://bookdown.org/yihui/rmarkdown-cookbook/.