Lecture 4: Panel Data

David Shilane

What We've Seen So Far

- Working on projects in data science is not easy. There are a lot of pieces to the puzzle.
- Each component – understanding the domain, investigation of the data, cleaning, analysis, reporting, and often other aspects of the work – can require many steps. Nothing is guaranteed to be straightforward.
- Early predictions about the scope of the work can drastically underestimate the labor that is required to accomplish a task.

The Tools of Data Science Simplify the Work

- Programming in R can greatly reduce the labor of processing and analyzing data.
- The methods of reproducible research and reporting can simplify the process of creating and editing reports. This all occurs with opportunities to improve the accuracy, transparency, and productivity of your work.
- The paradigms of computer science and software design enhance your capacity to create readable, modular, reusable, and high-performing code.

Experience Helps

- Each project you work on helps you to develop the habits that will carry into your future work.
- What you gain is different each time – some projects will teach you more about the methods, or writing code, or the challenges of a particular domain.
- In a new setting, your previous experience can still help you. Drawing analogies to aspects of your earlier work can be an effective way to develop the next project – and it can help you communicate those ideas to others even more.

Sticking to the Principles

- There are many ways to develop projects, write code, solve problems, communicate results, etc.

- The principles that I advocate for software design, reproducibility, and investigation of the data can be hard to live up to. They certainly require some upfront investment and a period of development. Even with a lot of practice, not all of my projects are perfectly aligned with my own guidelines! But it's still important to strive for success.

- Committing to good habits pays off immensely over time. **Coding well is no more difficult than coding poorly**, even in the moment.

It Gets Easier

- With good habits and experience, you'll notice that developing projects becomes easier.
- Fixing mistakes will occupy less of your time.
- Most importantly, your work can be more focused on understanding the fundamental problems you're trying to solve.

But Then It Becomes More Challenging Again

There are numerous ways in which projects in data science can becoming increasingly complex:

- Massive amounts of data: increasingly so
- Not the ideal form or quality of data: seemingly always
- Imperfect systems: ever more so, especially as the other components scale.

Different Structures of Data

- Data sets arrive in every shape and form.
- This is true **even after you have completed the work of cleaning and organizing the information**.
- Not every structure is so simple to work with.

Standard Format

I like to say that a set of data is in **standard format** (which is not necessarily the standard name) if it is structured to be as simple as possible to analyze:

- All of the rows of data follow the statistical properties of being **i.i.d.** – independent and identically distributed.
- There is one row per relevant observation (e.g. for a unique subject).
- Most of the standard analyses of the data are accessible in a minimal amount of work.

Everything We've Seen So Far

- The examples of data from the earlier lectures have all been in a standard format.
- Most analytical computations are as simple as calling the appropriate function, e.g. **sum**, **mean**, **sd**, etc.
- Even a computation that requires filtering and grouping can often be achieved in one step using our **data.table** methods.

Example: Diamonds Data

```
library(ggplot2)
data(diamonds)
dat <- as.data.table(diamonds)
dat[, .(N = .N, `Mean Price` = mean(price)), by = cut]
```

```
          cut     N Mean Price
1:      Ideal 21551   3457.542
2:    Premium 13791   4584.258
3:       Good  4906   3928.864
4: Very Good 12082   3981.760
5:       Fair  1610   4358.758
```

Standard Format Is Great

- We can very quickly explore, visualize, summarize, and analyze the data.
- Each step can be as simple as 1 line of code – or a relatively small number of them.
- In fact, the goal of our data cleaning work is to enable these kinds of simplified analyses. With clean data in standard format, you can answer numerous questions with relatively little work!

However, Standard Format Is Not Universal

Some data sets have a variety of reasons why it is not possible to work with a standard format.

- Fundamentally different kinds of data collection practices were useds.
- New variables were incorporated in the middle of the process.
- Different amounts of data are collected for different subjects.

Most Databases are not in Standard Format

- The information is spread out over many separate and loosely linked tables.
- New records are collected in real time.
- Some subjects produce many records, while others have very few.

Longitudinal Data

- Records collected over time for the same subject are often not independent, identically distributed observations.
- Not every subject always has records collected at the same time points.
- The range of experiences become increasingly wide over time.

Longitudinal Medical Records

Patients are followed over time. Updates are made as they come in:

- New diagnoses
- Laboratory tests
- Prescriptions
- Adverse events

There may also be gaps in the record:

- Changing Insurance Plans
- Updates in other systems that don't share their records

Panel Data /long-form time-varying data

One way to record longitudinal patient data is using *long-form time-varying* data, which is also known as *panel data*. It looks something like this:

Show 10 entries

period of time

Search:

	id	time1	time2	ACE	BB	statin	hospital	heart.attack
1	WIfCpBuK	0	30	1	0	1	1	1
2	WIfCpBuK	30	90	1	1	1	0	0
3	pGfvFIzo	0	14	0	1	0	1	0
4	pGfvFIzo	14	30	0	0	0	0	0
5	pGfvFIzo	30	60	1	1	1	1	1
6	ehjAEwKM	0	180	0	0	1	1	1
7	ehjAEwKM	180	365	1	1	0	0	0

Showing 1 to 7 of 7 entries

Previous 1 Next

Hallmarks of Panel Data

- Multiple Records Per Subject: Each subject is tracked over time. Different records appear in different rows of the data set.
- **id**: This is the subject's unique identifier. This is the way to link a patient's records together.
- **time intervals**: Each record pertains to a panel, which refers to an interval of time with a beginning (**time1**) and end (**time2**).
- **Baseline**: Time 0 is the beginning of our observation of the patient. This baseline may be when the diagnosis of a particular disease occurs.

Constant and Time-Varying Variables in Panel Data

- **Constant Variables**: Some variables have values that remain constant across the entire range of follow-up. This could include the patient's birthdate, eye color, or other factors that do not vary over time. For a single patient's records, these values will be the same in each row.

- **Time-Varying Variables**: Other variables have measurements that change over time, like the each patient's weight, current medications, or blood pressure. For a single patient, these values may change at any update.

- In some circumstances, a time-varying factor can actually be fixed as a constant variable. For instance, a patient's age **at the time of the first diagnosis of a medical condition** might be an interesting feature to note.

baseline age. eg:53 does change even 3 years later

Creating New Records

- When any change occurs for a single patient, a new row of data is created.
- All updates are assumed to take effect **at the beginning of the interval**. If a new row indicates that the patient is now taking beta blockers (BB), then this row will have the value 1.
- The condition in effect at the beginning of a time interval is **assumed to remain in effect until the end of the interval** for that row.

If a patient is listed as taking beta blockers from 0 to 30 days, then we can assume the patient was actively taking the medicine at any day in between.

Time Interval Assumptions

- The interval for a record from **time1** to **time2** is **includes time1** but **excludes time2**.
- The patient's next record could start as early as **time2** of the earlier row.
- Each row for each patient **must be mutually exclusive**. There can be no overlapping time intervals in separate rows.

Interpreting Panel Data

Show 10 entries Search:

	id	time1	time2	ACE	BB	statin	hospital	heart.attack
1	WIfCpBuK	0	30	1	0	1	1	1
2	WIfCpBuK	30	90	1	1	1	0	0
3	pGfvFIzo	0	14	0	1	0	1	0
4	pGfvFIzo	14	30	0	0	0	0	0
5	pGfvFIzo	30	60	1	1	1	1	1
6	ehjAEwKM	0	180	0	0	1	1	1
7	ehjAEwKM	180	365	1	1	0	0	0

Showing 1 to 7 of 7 entries Previous 1 Next

computer think death happens at the begining of the data
solution: create a new record 365->366, change the death sign

Patient WIfCpBuK was hospitalized with a heart attack at baseline. The patient was placed on an ACE inhibitor and a statin. At 30 days, the patient left the hospital and started taking a beta blocker. This regimen continued until day 90, when we stopped following the patient.

Advantages of Panel Data

- No set limits on time frames.
- Better reflects the dynamic changes of health.
- Adds rows instead of columns, which simplifies subsetting and extraction.
- Provides complete information based on what's available for individual patients.

richer than what you can get from the std dataset

Drawbacks of Panel Data

- The data are not in standard format. Many simple tools of analysis are either impractical or require augmentation.

Question: mean age of the patients?
can not just run mean function

- Big Data: A single patient could accrue a large volume of records over time, and the overall cohort might have a volume many times the sample size of patients might suggest.

#records>>#patients, be prepared to work with large records

- The sample size changes over time. Some patients are followed for longer than others.

Let's explore each of these points in greater detail.

Panel Data are not in Standard Format

- What does it mean to compute the average age of the patient? The percentage who are adherent to medications? The total number of hospitalizations?
- We cannot simply compute the **sum**, **mean**, or **standard deviation** of a variable across all of the rows. Some patients have more updates than others, thereby giving them disproportionate weight.
- Every simple calculation now has to **account for time**. This can mean different things depending on the context.

Summarizing Variables with Time in Mind

- **Mean age**: Rather than the overall average across all rows, we could compute the **average age at a specific time point**. This could be a calendar date or a **relative date**, like the average age of the patients **at the time of their initial diagnosis**.

- **Percentage adherent to beta blockers**: This would also need a time frame in mind. What percentage of patients were taking beta blockers 30 days after their initial diagnoses of heart disease? Or what percentage of the first year were patients taking beta blockers, averaged across all of the patients followed for a year?

- **Number of hospitalizations**: This would also need to be relative to time. How many total hospitalizations were there per patient per year when we restrict the analysis to the first year of follow-up?

Big Data

- Each patient could potentially accrue a large number of records.
- The more granular the information, the more records result. For instance, daily tracking of medication usage would create more updates than a quarterly record of purchases at the pharmacy.
- Example: A study I worked on had about 65,000 patients who were followed for up to 8 years. The resulting data set had about 2 million rows (about 30 per patient) and 100 columns. Back then, this was considered a large data set that required extra processing time.

Changing Sample Sizes

- Most analyses **establish a baseline for each subject**. This can be the moment that a user creates an account, a patient is diagnosed with a condition, or when an experiment begins.

- After the baseline, the number of subjects will begin to reduce. Over time, fewer subjects will remain under observation.

- Smaller sample sizes at later points in follow-up can mean less of an opportunity to draw conclusions. A large early sample size may not lead to enough remaining subjects at a future point to answer the question you had in mind.

Panel Data's Most Challenging Factor

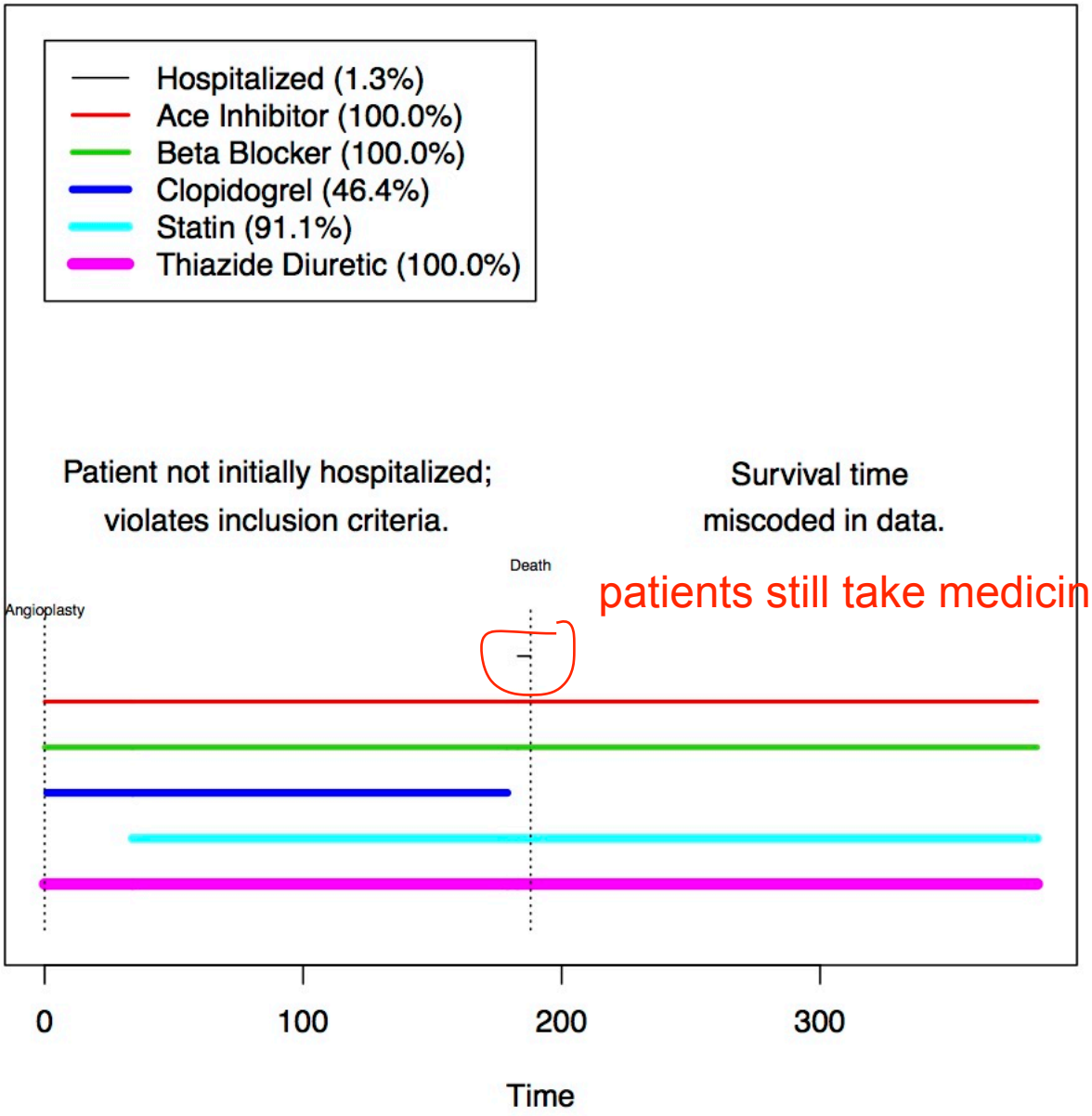
Panel data's volume and complexity of information will require extra care and attention in data cleaning!

Inspection is Crucial to Understanding Your Data

- It always make sense to give a closer look to some cases.
- For medical patients, ask if their story checks out.
- Visualizations can help you catch details you might otherwise miss.

A Truly Messy Data Set

Medical History for Subject 5009166832



patients still take medicines after death???

Investigation Rescued the Project

- For patients who died, the last record combined the prior update and their death.
- However, the assumptions of our calculations and models were that **all updates occur at the beginning**. That is, we assumed an earlier time of death than what truly occurred.
- This was easy to fix: We added another row for each death and updated the records accordingly.
- Ultimately these data formed the basis for multiple medical studies. Without discovering this error, all of these studies **would have been based on faulty data**.

Panel Data Can Hone Your Abilities

- Learning to operate with a more complex structure will help you develop new skills.
- You'll also think about problems differently, which will help you adapt to any new structure that you encounter in your next project.
- Panel data's longitudinal structure forces us to ask questions that are more temporal in nature.

Deep Dive: Diabetes Panel Data

- We will investigate the information contained in the file **simulated diabetes panel data.csv**, which is available on the course website.
- We will seek to answer some simple questions about the health of patients with diabetes.
- Along the way, we'll learn techniques for working with panel data.

Study Design

- A medical organization is testing out a **new intervention** that provides a series of three health and wellness classes to patients who are newly diagnosed with Type II diabetes mellitus. These classes are given at no charge to the patient in addition to the standard medical care.
- As a **control group**, other patients with newly diagnosed Type II diabetes mellitus will only receive the standard medical care.
- The patients are **randomly assigned** to the treatment and control cohorts.

Outcomes to Compare

- Rates of medication adherence will be tracked and compared.
- Comparative reductions in the weight and A1C scores of the patients will also be of interest.
- The two cohorts will be compared in terms of their rates of hospitalization over time.

Reading in the Diabetes Panel Data

```
library(DT)
library(data.table)
dat <- fread(input = "simulated diabetes panel data.csv")
datatable(data = dat[1:20, ])
```

Show 10 entries Search:

	id	t1	t2	intervention	age	weight	a1c	metformin	statin	hospit
					age been diago					
1	km12ncxA	381	399	0	38	128.2	9.9	0	0	
2	xRcUqOqR	222	237	1	38	130.5	9	0	1	
3	km12ncxA	363	381	0	38	131.4	9.9	0	0	
4	xRcUqOqR	199	222	1	38	132.8	8.8	0	1	
5	xRcUqOqR	172	199	1	38	133.3	8.8	0	0	
6	km12ncxA	338	363	0	38	134.7	9.9	0	0	
7	xRcUqOqR	138	172	1	38	135.9	8.7	0	0	
8	xRcUqOqR	105	138	1	38	137.8	8.9	0	0	
9	km12ncxA	302	338	0	38	139.2	9.9	0	0	

	id	t1	t2	intervention	age	weight	a1c	metformin	statin	hospit
10	xRcUqOqR	79	105	1	38	140.3	9.1	0	0	

Showing 1 to 10 of 20 entries

Previous 1 2 Next

Describing the Variables:

- **id**: A unique identifying string for each patient. The **id** separately links to the patient's name and information about the account. No information that could reasonably identify a patient should be included in this study.

- **t1**: The beginning of a time interval. This is measured in days. The time measures is the number of days since the patient's initial diagnosis of Type II diabetes.

- **t2**: The ending of a time interval. This is measured in days. The time measures is the number of days since the patient's initial diagnosis of Type II diabetes.

- **intervention**: Whether the patient received the intervention (1) or not (0).

- **age**: This is the age of the patient in years **at the time of the initial diagnosis of diabetes**.

More Variables

- **weight**: This is the patient's measured weight in pounds at the specified point of time.
- **a1c**: This is the patient's percentage of **glycated hemoglobin** at the specified time point. Patients are typically diagnosed with Type II diabetes if this value is at least 6.5. Higher scores are progressively worse for a patient's overall health.
- **metformin**: This is an indicator of whether the patient has possession of the prescribed medication metformin (1) or not (0) at the specified time point.
- **statin**: This is an indicator of whether the patient has possession of the prescribed medication statin (1) or not (0) at the specified time point.
- **hospital**: This is an indicator of whether the patient was in the hospital (1) or not (0) at the specified time point.

Initial Questions

- How many rows of data are we working with?

```
dat[, .N]
```

```
[1] 111327
```

- How many unique patients are included?

probably 11 records/patient

```
id.name <- "id"
dat[, length(unique(get(id.name)))]
```

```
[1] 10000
```

Examining One Patient's Records

- Let's take a look at all of the records for a single patient:

```
the.id <- dat[1, get(id.name)]
datatable(data = dat[get(id.name) == the.id, ], rownames = FALSE)
```

Show 10 entries Search:

id	t1	t2	intervention	age	weight	a1c	metformin	statin	hospital
km12ncxA	381	399	0	38	128.2	9.9	0	0	0
km12ncxA	363	381	0	38	131.4	9.9	0	0	0
km12ncxA	338	363	0	38	134.7	9.9	0	0	0
km12ncxA	302	338	0	38	139.2	9.9	0	0	0
km12ncxA	266	302	0	38	142.9	9.8	1	0	0
km12ncxA	237	266	0	38	145.2	9.8	0	0	0
km12ncxA	215	237	0	38	148.6	9.8	0	1	0
km12ncxA	183	215	0	38	151.8	9.8	0	1	0

id	t1	t2	intervention	age	weight	a1c	metformin	statin	hospital
km12ncxA	168	183	0	38	153.8	10	0	0	0
km12ncxA	151	168	0	38	155.3	9.9	0	1	0

Showing 1 to 10 of 16 entries

Previous 1 2 Next

Reordering the Table

- Overall, the full table **dat** appears to be ordered by age and weight.
- It would make more sense to order the data by **id** and then the beginning of the time period **t1**:

```
t1.name <- "t1"
setorderv(x = dat, cols = c(id.name, t1.name))
datatable(data = dat[1:20, ], rownames = FALSE)
```

Show 10 entries Search:

id	t1	t2	intervention	age	weight	a1c	metformin	statin	hospital
008sOcDq	0	35	0	66	198.4	10.7	1	0	0
008sOcDq	35	57	0	66	197.6	10.4	1	0	0
008sOcDq	57	79	0	66	195.9	10.3	1	1	0
008sOcDq	79	96	0	66	195.3	10.4	0	1	0
008sOcDq	96	122	0	66	194.7	10.6	1	1	0
00gAD16F	0	33	0	53	212.6	10.4	0	1	0

id	t1	t2	intervention	age	weight	a1c	metformin	statin	hospital
00gAD16F	33	53	0	53	215.2	10.4	0	0	0
00gAD16F	53	95	0	53	215.4	10.4	0	1	0
00gAD16F	95	123	0	53	215.2	10.4	1	1	0
00gAD16F	123	127	0	53	216.5	10.6	1	1	1

Re-Examining One Patient's Records

- Let's take another look at the first patient we examined, now with ordered records:

```
datatable(data = dat[get(id.name) == the.id, ], rownames = FALSE)
```

Show 10 entries Search:

id	t1	t2	intervention	age	weight	a1c	metformin	statin	hospital
km12ncxA	0	44	0	38	173.3	9.9	0	0	0
km12ncxA	44	49	0	38	168.4	9.9	0	0	1
km12ncxA	49	62	0	38	168.4	9.9	0	0	0
km12ncxA	62	83	0	38	165.9	9.9	0	0	0
km12ncxA	83	108	0	38	163.6	10	0	0	0
km12ncxA	108	151	0	38	160.4	10	0	1	0
km12ncxA	151	168	0	38	155.3	9.9	0	1	0
km12ncxA	168	183	0	38	153.8	10	0	0	0

id	t1	t2	intervention	age	weight	a1c	metformin	statin	hospital
km12ncxA	183	215	0	38	151.8	9.8	0	1	0
km12ncxA	215	237	0	38	148.6	9.8	0	1	0

The Patient's History

This patient was randomized not to receive the intervention. A total of 16 records were collected over a period of 399 days. The patient was 38 years old, reduced weight from 173.3 to 128.2 pounds during follow-up. The A1C scores were in the range from 9.8 to 10. The patient barely used metforming and was intermittently on a statin. The patient also had 1 hospitalization in follow-up.

Examining the Time Fields

- Let's begin the investigation by taking a closer look at the time periods in question. There are a number of simple checks of quality to consider:

- There are no negative time periods:

```
t2.name <- "t2"
dat[get(t1.name) < 0 | get(t2.name) < 0, .N]
```

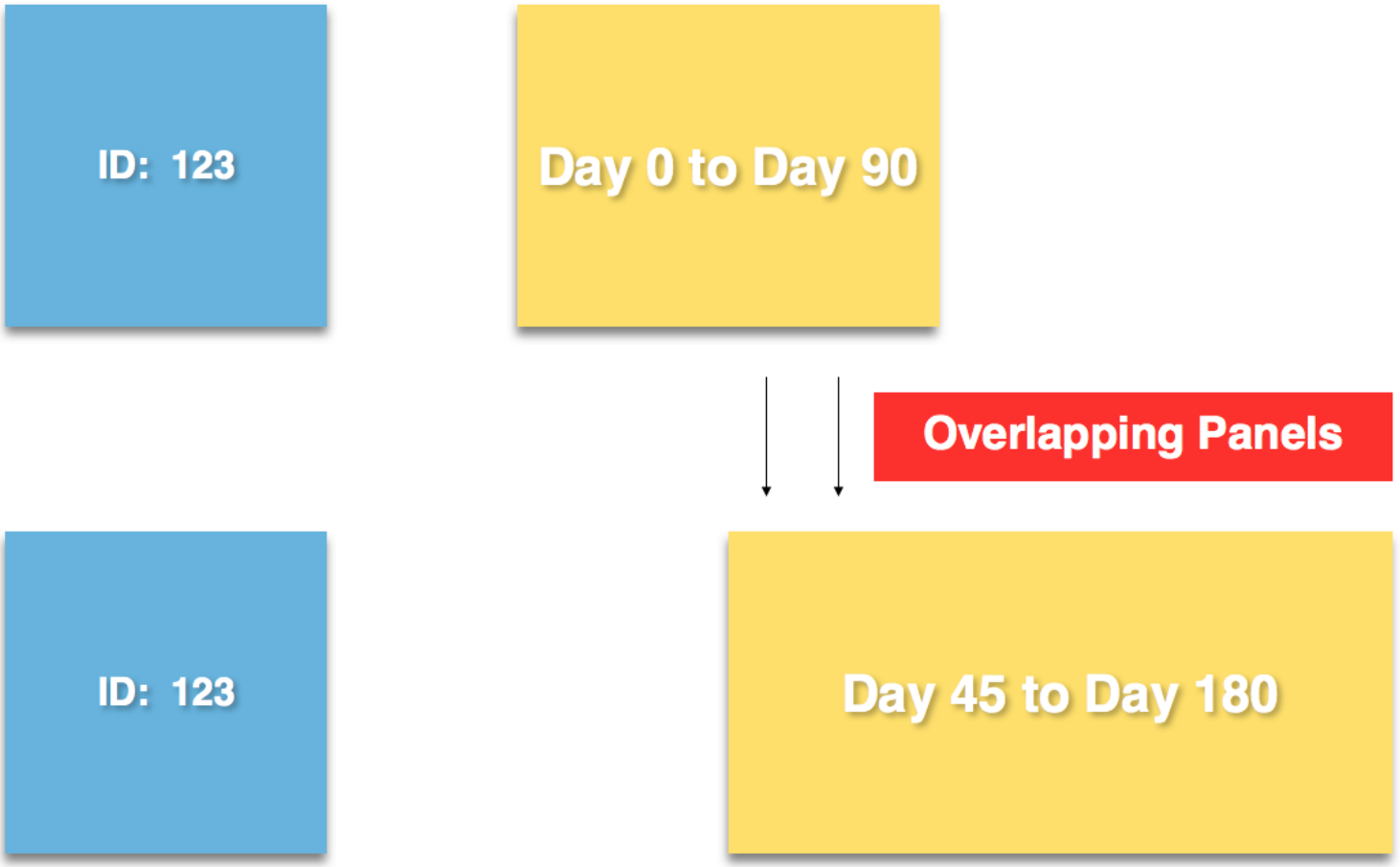
```
[1] 0
```

- There are no beginning times that are greater than the corresponding ending times:

```
dat[get(t1.name) >= get(t2.name), .N]
```

```
[1] 0
```

Overlapping Time Frames



A Plan for Investigation

- No patient can have two records with overlapping time frames; each period of time must be mutually exclusive. How do we check for this? Let's create a plan:

- We'll write a function to check **a single patient's records**. This will investigate whether the time frames in one row overlap those in another.

- Then, we'll write another function that performs this task on each individual patient and **aggregates the results**.

Investigating the Time Frames of One Patient

```
id.panel.overlaps.one.patient <- function(patient.dat, id.name, t1.name, t2.name,
    row.index.name) {
    require(data.table)
    setDT(patient.dat)

    beginning.times <- patient.dat[, get(t1.name)]
    ending.times <- patient.dat[, get(t2.name)]

    overlapping.results <- patient.dat[, .(has_overlap = sum((get(t1.name) >
        beginning.times & get(t1.name) < ending.times) | (get(t2.name) < beginning.times &
        get(t2.name) > ending.times)) > 0), by = row.index.name]
    overlapping.true.false <- overlapping.results[, sum(has_overlap) > 0]

    return(overlapping.true.false)
}
```

Extending to All of the Patients

```
id.panel.overlaps <- function(dat, id.name, t1.name, t2.name){
  require(data.table)
  setDT(dat)
  setorderv(x = dat, cols = c(id.name, t1.name), order = 1)
  dat[, record.index := 1:.N, by = id.name]

  ids.with.overlaps <- dat[, .(V1 = id.panel.overlaps.one.patient(patient.dat = .SD, id.name = id.name, t1.name =
  t1.name, t2.name = t2.name, row.index.name = "record.index")), by = get(id.name)]

  setnames(x = ids.with.overlaps, old = c("get", "V1"), new = c(id.name, "overlapping_panels"))
  return(ids.with.overlaps)
}
```

```
overlaps <- id.panel.overlaps(dat = dat, id.name = id.name, t1.name = t1.name,
    t2.name = t2.name)
overlaps[overlapping_panels == TRUE, .N]
```

```
[1] 0
```

But What If There Were Overlaps?

- The **overlaps** data.table we calculated would be able to identify the patients whose records had overlapping time frames.
- Some investigation of the data would be in order. What is the nature of the error, and how did it come about?
- This may require you to discuss the issue with the engineering team or other members of the organization.

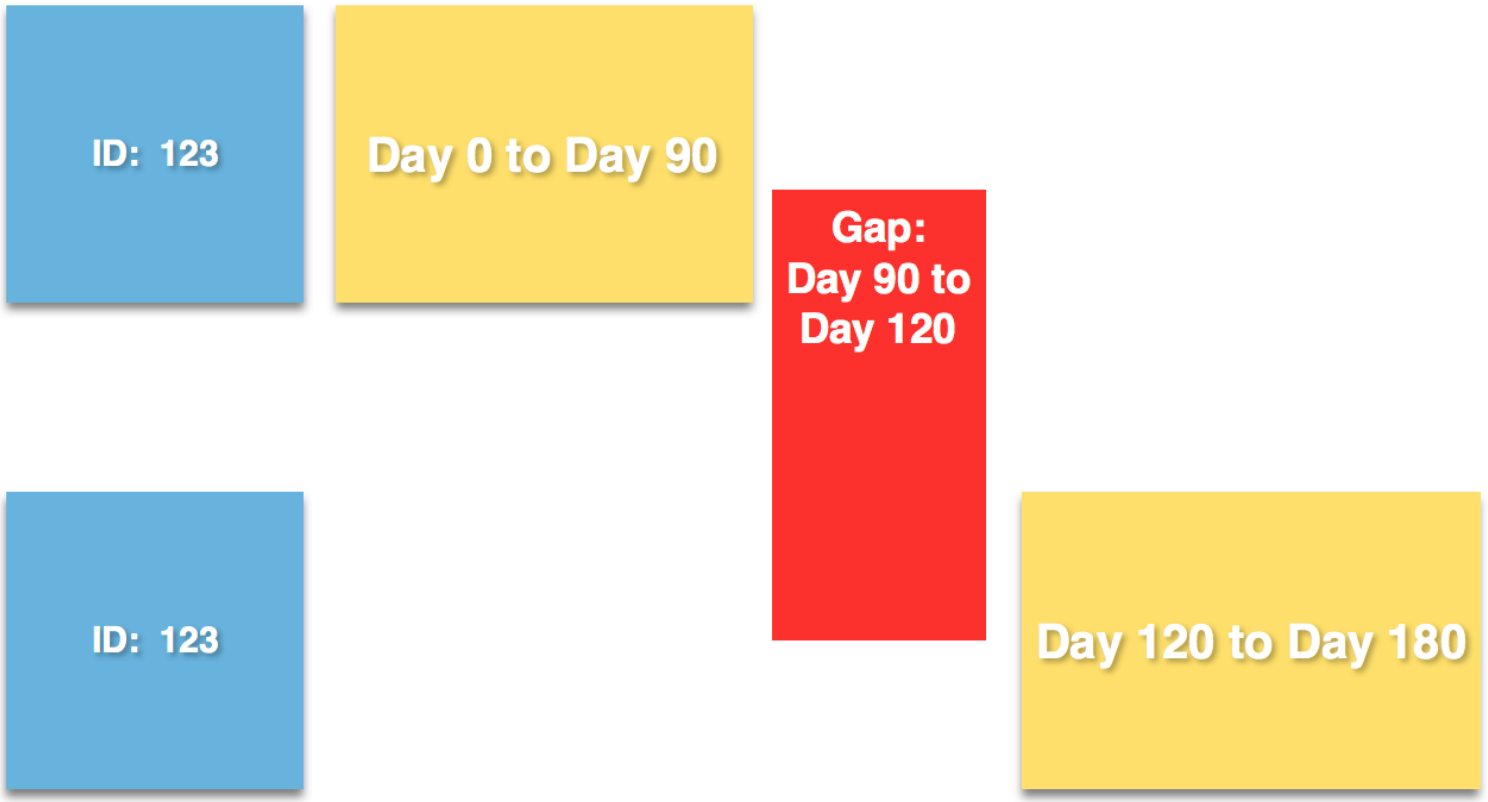
Creating Panel Data Is Challenging for Data Engineers

- Behind the scenes, an engineering team has to identify all of the relevant database entries, properly extract the information, and code it in a sensible manner.

- Then there is a process of constructing the panels (the time periods) based on the full list of recorded variables.

- The most common mistake in this process **is adding a new variable** without fully re-applying the generating code across the full range of measures. Because each panel is constructed **when any single change takes place**, it is necessary to perform this computation in a comprehensive manner. There are no shortcuts.

Gaps in Time Intervals



Missing Data

- The other serious problem is that not every time period is necessarily included over the range of follow-up. There may be **gaps** of time for which no records are recorded on the patient.

- Why might this be? Sometimes there is a **loss of follow-up** for a period of time. When a patient moves away temporarily or switches to a new insurance plan, the will be gaps.

- Additionally, time periods can be subject to missing records just like any other variable. The information might not be recorded for any number of reasons.

Identifying Gaps – One Patient

```
## Assumes that the data are all for a single patient (same id) and sorted in
## increasing order of t1.name
identify.panel.gaps.one.patient <- function(patient.dat, t1.name, t2.name, first.value = 0,
    expected.gap.between = 0) {
    require(data.table)
    setDT(patient.dat)

    gap.first.row <- (patient.dat[1, get(t1.name) > first.value])
    n <- patient.dat[, .N]

    if (n == 1) {
        res <- gap.first.row
    }
    if (n > 1) {
        t2.values <- patient.dat[1:(n - 1), get(t2.name)]
        gaps.other.rows <- patient.dat[2:n, get(t1.name) > t2.values + expected.gap.between]
        res <- c(gap.first.row, gaps.other.rows)
    }
    return(res)
}
```

Identifying Gaps – All Patients

```
identify.panel.gaps <- function(dat, id.name, t1.name, t2.name, gap.name = "gap_before",
    first.value = 0, expected.gap.between = 0) {
    require(data.table)
    setDT(dat)
    setorderv(x = dat, cols = c(id.name, t1.name), order = 1)

    dat[, `:=`(eval(gap.name), identify.panel.gaps.one.patient(patient.dat = .SD,
        t1.name = t1.name, t2.name = t2.name, first.value = first.value, expected.gap.between =
  expected.gap.between)),
        by = get(id.name)]

    return(dat[])
}
```

3

28

GAP

30

60

Counting Gaps

- How many rows reveal gaps in the data?

```
gap.name = "gap_before"
dat <- identify.panel.gaps(dat = dat, id.name = id.name, t1.name = t1.name,
    t2.name = t2.name, gap.name = gap.name)

total.gaps <- dat[get(gap.name) == TRUE, .N]
tab <- dat[, .(`Number of Gaps` = total.gaps, `Number of Rows` = .N, `Percentage of Gaps` = 100 *
    total.gaps/.N)]
datatable(data = tab[, lapply(X = .SD, FUN = "round.numerics", digits = 2)],
    rownames = FALSE)
```

Show 10 entries Search:

Number of Gaps	Number of Rows	Percentage of Gaps
1924	111327	1.73

Showing 1 to 1 of 1 entries Previous 1 Next

Impact of the Gaps

- What percentage of days are lost due to measurement gaps?

```
total.days <- dat[, .(max_measured = max(get(t2.name))), by = id.name][, sum(max_measured)]
days.lost <- dat[, total.days - sum(get(t2.name) - get(t1.name))]
tab.lost <- data.table(`Possible Days` = total.days, `Measured Days` = total.days -
    days.lost, `Days Lost` = days.lost, `Percentage Lost` = 100 * days.lost/total.days)
datatable(data = tab.lost[, lapply(X = .SD, FUN = "round.numerics", digits = 2)],
    rownames = FALSE)
```

Show 10 entries Search:

Possible Days	Measured Days	Days Lost	Percentage Lost
2996057	2944464	51593	1.72

Showing 1 to 1 of 1 entries Previous 1 Next

What We Can't Know

- There may be additional gaps **after the last record** for the patient.
- Any such gap could be the result of either **missing data** or from data that was **never collected** due to a loss of follow-up.
- You may be able to track this down through your organization's quality assurance and review processes. But, most of the time, **you can't really know** if there should have been more follow-up data.

Baseline Measurements

- There are a variety of factors to investigate about the beginning (or **baseline**) of the intervention.
- The starting condition of the patients can be used to assess the risk of longitudinal outcomes like hospitalization, track the progression of factors like weight, and examine the shorter-term questions about how newly diagnosed patients are managed.
- But to do this, we need to think carefully about the baseline and what we're measuring.

But We Have Missing Data

- The baseline should occur at day 0. However, some patients have missing data. What is the impact here?

```
dat[, .(`Total Patients` = length(unique(get(id.name))))]
```

```
   Total Patients
1:          10000
```

```
dat[get(t1.name) == 0, .(`Total Patients at Baseline` = length(unique(get(id.name))))]
```

```
   Total Patients at Baseline
1:                       9906
```

Different Choices

- **Approach 1**: You could decide that the baseline measurement only includes those patients with a measured value at day 0.
- **Approach 2**: You could also decide that the **first measured value** for each patient is *close enough* to the baseline.
- **Approach 3**: You could also choose to **backfill** the missing values through some means. This could include **extrapolation** of the first measured result (Approach 2) or **imputation methods** that range from simple (averaging) to complex (multivariable predictions or machine learning). The key difference to the Approach 3 is that you would be **adding rows** for the patients with missing values.

We will briefly examine these choices with regard to counting the number of patients in each cohort at baseline.

Counting Interventions

- We want to know how many unique patients were in the treatment cohort (receiving the intervention's health classes plus standard care) or the control group (standard care).
- This is good to track at various time points, most of all at baseline.
- We'll provide counts using the 3 approaches to handling the missing data.

Counting Interventions: Different Approaches

- How many unique patients were in each cohort with measured values at day 0?

```
intervention.name <- "intervention"
dat[get(t1.name) == 0, .(`Number of Patients` = length(unique(get(id.name)))),
    by = intervention.name]
```

```
   intervention Number of Patients
1:            0                4967
2:            1                4939
```

- How many unique patients were in each cohort **at their first measured value**?

```
setorderv(x = dat, cols = c(id.name, t1.name), order = 1)
first.rows <- dat[, .SD[1], by = id.name]   get the first row per person
first.rows[, .(`Number of Patients` = length(unique(get(id.name)))), by = intervention.name]
```

```
   intervention Number of Patients
1:            0                5014
2:            1                4986
```

data.table's .SD Functionality

- Recall that using **.SD** in the **j step** of a calculation on a data.table selects all of the columns listed in the **.SDcols** argument (all columns by default).
- The line **dat[, .SD[1]]** selects **the entire first row** of the data.table dat. The line **dat[, .SD[1], by = id.name]** selects **the entire first row for each unique id** in dat.
- When the data.table is sorted by **id** and **t1**, then it allows us to select **the first measured value** for each patient. This is especially helpful because we do not have to impose any constraints on what the time frame might be.

Approach 3: Identification of Cases

- Using imputation or extrapolation to add new rows to the data requires identifying the patients with missing baseline records.

```
missing.baseline <- first.rows[get(t1.name) != 0, ]
datatable(data = missing.baseline, rownames = FALSE)
```

Show 10 entries Search:

id	t1	t2	intervention	age	weight	a1c	metformin	statin	hospital
06AcgrhR	33	63	0	56	267	11.6	1	1	0
0WLYY51c	6	28	0	63	222.5	10.4	0	0	0
1d8gdmtu	40	76	1	40	230.5	10.7	1	1	0
1wgVDa4T	35	59	0	44	199	10.3	1	1	0
2LsdgEvj	29	45	1	51	204	10.4	0	1	0
2egTf5ul	26	63	1	67	202.5	9.2	0	0	0
348mRimp	21	63	1	46	218.4	10.6	1	0	0
39lW9kvV	31	70	1	56	207.4	10.7	1	1	0

id	t1	t2	intervention	age	weight	a1c	metformin	statin	hospital
3AG6HCtK	27	65	1	50	190.3	10.1	1	1	0
3OH1H6aB	39	62	0	53	226.9	10.5	1	0	0

Showing 1 to 10 of 94 entries

Previous 1 2 3 4 5 … 10 Next

Approach 3: Extrapolation

- Then, using extrapolation, we can create additional rows by setting **t1** to 0 and **t2** to the previous value of **t1**:

```
setnames(x = missing.baseline, old = c(t1.name, t2.name), new = c(t2.name, "to.remove"))
missing.baseline[, eval(t1.name) := 0]
missing.baseline[, to.remove := NULL]
datatable(data = missing.baseline, rownames = FALSE)
```

Show 10 entries Search:

id	t2	intervention	age	weight	a1c	metformin	statin	hospital	reco
06AcgrhR	33	0	56	267	11.6	1	1	0	
0WLYY51c	6	0	63	222.5	10.4	0	0	0	
1d8gdmtu	40	1	40	230.5	10.7	1	1	0	
1wgVDa4T	35	0	44	199	10.3	1	1	0	
2LsdgEvj	29	1	51	204	10.4	0	1	0	
2egTf5ul	26	1	67	202.5	9.2	0	0	0	
348mRimp	21	1	46	218.4	10.6	1	0	0	

id	t2	intervention	age	weight	a1c	metformin	statin	hospital	reco
39lW9kvV	31	1	56	207.4	10.7	1	1	0	
3AG6HCtK	27	1	50	190.3	10.1	1	1	0	
3OH1H6aB	39	0	53	226.9	10.5	1	0	0	

Showing 1 to 10 of 94 entries

Previous 1 2 3 4 5 … 10 Next

Incorporating the New Rows

- Note that the backfilling operation only pertained to the missing baseline values. We did not do this for the other gaps that occurred during follow-up.

- Then, we can bind in the extrapolated version of the missing rows to the original data:

```
backfilled.dat <- rbindlist(l = list(dat, missing.baseline), fill = TRUE)
setorderv(x = backfilled.dat, cols = c(id.name, t1.name), order = 1)
backfilled.dat[get(t1.name) == 0, .(`Number of Patients` = length(unique(get(id.name)))),
    by = intervention.name]
```

```
   intervention Number of Patients
1:            0                5014
2:            1                4986
```

How to Proceed

- The choices you make about handling the missing baseline data should incorporate the computational needs of your team, an investigation about the quality of the information, and any other considerations that will be important for the project.

- My own preference would be to stick with the original information for as long as you can. Only start making fundamental changes if you really need to.

- For now, we will proceed with Approach 2 – using the first record for each patient – when questions about the baseline are addressed.

Investigating Age

- Age is supposed to be the patient's age in years (floored) at baseline. For each patient, the value of age should be constant across their records.

- This is worth investigating:

```
age.name <- "age"
age.uniqueness.counts <- dat[, .(`Unique Age Values` = length(unique(get(age.name)))),
    by = id.name]
age.uniqueness.counts[, mean(`Unique Age Values` == 1)]
```

```
[1] 1
```

- With that consistency ensured, we can summarize the baseline age:

```
first.rows[, summary(get(age.name))]
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  38.00   47.00   55.00   55.32   64.00   72.00
```

Differences in Age by Cohort at Baseline

- Comparing the **mean** and **sd** of age in each cohort:

```
mean.and.sd <- function(x, x.name) {
    require(data.table)
    res <- data.table(V1 = mean(x, na.rm = TRUE), V2 = sd(x, na.rm = TRUE))
    setnames(x = res, old = c("V1", "V2"), new = sprintf("%s of %s", c("Mean",
        "SD"), x.name))
    return(res)
}
```

```
age.tab.by.cohort <- first.rows[, mean.and.sd(x = get(age.name), x.name = eval(age.name)),
    by = intervention.name]
datatable(data = age.tab.by.cohort[, lapply(X = .SD, FUN = "round.numerics",
    digits = 2)], rownames = FALSE)
```

Show 10 entries Search:

intervention	Mean of age	SD of age
0	55.26	10.14
1	55.38	10.11

Showing 1 to 2 of 2 entries Previous 1 Next

- These values look quite comparable, as should be expected from a randomized trial. Performing a statistical test (like a t-test) would also be advised to assess the significance of the difference in means.

Time-Varying Information

- Intervention and Baseline Age were measured as constant variables from the baseline information.
- Weight, A1C, Metformin usage, Statin usage, and Hospitalizations were time-varying factors that could change with each new record.
- Each variable has its own associated questions, methods of analyses, and results to report.

So let's dig in on each one.

Cross-Sectional Data

- Let's create data.tables with one row for each (measured) patient at baseline, 3 months, and 6 months:

```
cross.sectional.data <- function(dat, time.point, t1.name, t2.name, id.name) {
    require(data.table)
    setDT(dat)
    setorderv(x = dat, cols = c(id.name, t1.name))
    cs.dat <- dat[get(t2.name) > time.point][, .SD[1], by = id.name]
    return(cs.dat)
}
one.year <- 365.25
baseline <- first.rows
d3mo <- cross.sectional.data(dat = dat, time.point = one.year/4, t1.name = t1.name,
    t2.name = t2.name, id.name = id.name)
d6mo <- cross.sectional.data(dat = dat, time.point = one.year/2, t1.name = t1.name,
    t2.name = t2.name, id.name = id.name)
```

find the row right before the t1>time.point

Cross-Sectional Summaries of Weight

- How did the population's weight change over time and by cohort? Show the mean and the standard deviation
- Let's produce these measures at the following time points: baseline, 3 months, and 6 months:

```
weight.name <- "weight"
weight.baseline <- data.table(Month = 0, baseline[, mean.and.sd(x = get(weight.name),
    x.name = weight.name), by = intervention.name])
weight.3mo <- data.table(Month = 3, d3mo[, mean.and.sd(x = get(weight.name),
    x.name = weight.name), by = intervention.name])
weight.6mo <- data.table(Month = 6, d6mo[, mean.and.sd(x = get(weight.name),
    x.name = weight.name), by = intervention.name])

tab.weight <- rbindlist(l = list(weight.baseline, weight.3mo, weight.6mo), fill = TRUE)
datatable(data = tab.weight[, lapply(X = .SD, FUN = "round.numerics", digits = 1)],
    rownames = FALSE)
```

Show 10 entries Search:

Month	intervention	Mean of weight	SD of weight
0	0	210.3	19.8
0	1	210.1	20.1

might t-test, regression model.ml to explore if apply the intervention make sense

Month	intervention	Mean of weight	SD of weight
3	0	207.7	20.1
3	1	207.1	20.2
6	0	204.7	21.6
6	1	203.2	21.4

Showing 1 to 6 of 6 entries

Previous 1 Next

might t-test, regression model.ml to explore if apply the intervention make sense

might t-test, regression model.ml to explore if apply the intervention make sense

Percentage Who Lose Weight

- Among those with measured values at both times, how many and what percentage of patients lost weight at 3 months relative to their baseline?
- Let's write a function to calculate this:

```
mean.with.reduction <- function(d1, d2, variable.name, id.name) {
    require(data.table)
    setDT(d1)
    setDT(d2)
    dat.merged <- merge(x = d1, y = d2, by = id.name, all = FALSE)

    first.name <- sprintf("%s.x", variable.name)
    second.name <- sprintf("%s.y", variable.name)
    mean.reduced <- dat.merged[, mean(get(first.name) > get(second.name), na.rm = TRUE)]
    return(mean.reduced)
}
```

Percentage Losing Weight at 3 and 6 Months

- 3 Months:

```
pct.weight.loss.3mo <- 100 * mean.with.reduction(d1 = baseline, d2 = d3mo, variable.name = weight.name,
    id.name = id.name)
pct.weight.loss.3mo
```

```
[1] 72.46157
```

- 6 Months:

```
pct.weight.loss.6mo <- 100 * mean.with.reduction(d1 = baseline, d2 = d6mo, variable.name = weight.name,
    id.name = id.name)
pct.weight.loss.6mo
```

```
[1] 71.78307
```

Percentage Losing Weight By Intervention

- 3 Months:

```
pct.weight.loss.3mo.treatment <- 100 * mean.with.reduction(d1 = baseline[get(intervention.name) ==
    1, ], d2 = d3mo[get(intervention.name) == 1, ], variable.name = weight.name,
    id.name = id.name)
pct.weight.loss.3mo.control <- 100 * mean.with.reduction(d1 = baseline[get(intervention.name) ==
    0, ], d2 = d3mo[get(intervention.name) == 0, ], variable.name = weight.name,
    id.name = id.name)

tab.weight.loss.pct.3mo <- data.table(Month = 3, `Percentage Losing Weight, Control` = pct.weight.loss.3mo.control,
    `Percentage Losing Weight, Treatment` = pct.weight.loss.3mo.treatment)
```

- 6 Months:

```
pct.weight.loss.6mo.treatment <- 100 * mean.with.reduction(d1 = baseline[get(intervention.name) ==
    1, ], d2 = d6mo[get(intervention.name) == 1, ], variable.name = weight.name,
    id.name = id.name)
pct.weight.loss.6mo.control <- 100 * mean.with.reduction(d1 = baseline[get(intervention.name) ==
    0, ], d2 = d6mo[get(intervention.name) == 0, ], variable.name = weight.name,
    id.name = id.name)

tab.weight.loss.pct.6mo <- data.table(Month = 6, `Percentage Losing Weight, Control` = pct.weight.loss.6mo.control,
    `Percentage Losing Weight, Treatment` = pct.weight.loss.6mo.treatment)
```

Weight Loss Results By Cohort

```
tab.weight.loss.pct <- rbindlist(l = list(tab.weight.loss.pct.3mo, tab.weight.loss.pct.6mo),
    fill = TRUE)
datatable(data = tab.weight.loss.pct[, lapply(X = .SD, FUN = "round.numerics",
    digits = 1)], rownames = FALSE)
```

Show 10 entries Search:

Month	Percentage Losing Weight, Control	Percentage Losing Weight, Treatment
3	70.7	74.3
6	69.5	74

Showing 1 to 2 of 2 entries Previous 1 Next

A1C Scores

- We could go through a similar line of reasoning to show the reductions in A1C scores.
- We could also look at the **mean percentage reduction** in scores from baseline to 6 months.
- Likewise, we could investigate whether patients reach certain thresholds.

Mean Percentage Reduction

- Each patient's score goes up or down over time. Rather than measuring reductions in absolute terms, we could work with the percentage by which each patient's score is reduced.
- This method more fairly accounts for different starting points in the scores.

```
mean.with.reduction <- function(d1, d2, variable.name, id.name, by) {
    require(data.table)
    setDT(d1)
    setDT(d2)
    dat.merged <- merge(x = d1, y = d2, by = id.name, all = FALSE)

    first.name <- sprintf("%s.x", variable.name)
    second.name <- sprintf("%s.y", variable.name)
    new.by.name <- sprintf("%s.x", by)

    mean.reduced <- dat.merged[, .(V1 = mean(get(first.name)/get(second.name),
        na.rm = TRUE)), by = eval(new.by.name)]
    setnames(x = mean.reduced, old = c(new.by.name, "V1"), new = c(by, sprintf("Mean %s Ratio",
        variable.name)))
    return(mean.reduced)
}
```

6 Month Mean Percentage Reduction in A1C by Cohort

```
a1c.name <- "a1c"
mean.a1c.reduction.by.cohort <- mean.with.reduction(d1 = baseline, d2 = d6mo,
    variable.name = a1c.name, id.name = id.name, by = intervention.name)
datatable(data = mean.a1c.reduction.by.cohort[, lapply(X = .SD, FUN = "round.numerics",
    digits = 3)], rownames = FALSE)
```

Show 10 entries Search:

intervention	Mean a1c Ratio
0	1.004
1	1.004

Showing 1 to 2 of 2 entries Previous 1 Next

- Because the ratios are close to 1, it does not appear that the average patient sees a reduction in A1C scores in 6 months, with or without the intervention.

Medication Analyses

Longitudinal information about filled prescriptions enables us to explore a variety of interesting questions:

- What percentage of the patients fill a prescription for each medicine within 30 days of baseline?
- On average, how long does it take for patients to get started on each medication?
- Over the first 6 months of follow-up, what percentage of the time does the average patient take each medicine?

Let's explore each of these questions in greater detail.

Starting within 30 Days

- A patient is classified as starting the medication if a prescription is filled within within 30 days.
- This remains true even if the patient later does not fill a prescription, even if that occurs during the 30-day time frame.
- For the patients with missing baseline records, we'll assert that it's OK to extrapolate from their first measured record. This would be a good time to use the **backfilled.dat** object created earlier.

Percentage Starting within 30 Days

```
the.threshold <- 30
metformin.name <- "metformin"
statin.name <- "statin"
meds <- c(metformin.name, statin.name)
meds.tab <- backfilled.dat[get(t1.name) < the.threshold, lapply(X = .SD, FUN = "max",
    na.rm = TRUE), .SDcols = meds, by = c(id.name, intervention.name)]
starting.pct <- meds.tab[, lapply(X = .SD, FUN = "mean", na.rm = TRUE), .SDcols = meds,
    by = intervention.name]
datatable(data = starting.pct[, lapply(X = .SD, FUN = "round.numerics", digits = 3)],
    rownames = FALSE)
```

Show 10 entries Search:

intervention	metformin	statin
0	0.624	0.702
1	0.608	0.697

Showing 1 to 2 of 2 entries Previous 1 Next

Starting Times

- Our original question was: **On average, how long does it take for patients to get started on each medication?**
- However, some patients **never** fill a prescription. How would we measure their time?
- It would make more sense to **restrict our attention** to the patients who fill a prescription **at least once in follow-up**.

Note: For this analysis, we'll once again use the **backfilled.dat**. However, that extrapolation impacts the average starting time! Keep that in mind.

Average Starting Time, Among Those Who Start

```
mean.starting.time <- function(all.dat, variable.name, t1.name, id.name, by, ever.used.name = "ever_used"){
  require(data.table)
  setDT(all.dat)
  all.dat[, eval(ever.used.name) := max(get(variable.name), na.rm = TRUE), by = c(id.name, by)]
  starting.times <- all.dat[get(ever.used.name) == 1 & get(variable.name) == 1, .(starting.time = min(get(t1.name))),
  by = c(id.name, by)]
  all.dat[, eval(ever.used.name) := NULL]
  start.tab <- starting.times[, .(V1 = mean(starting.time)), by = by]
  setnames(x = start.tab, old = "V1", new = sprintf("Mean Days to Start %s", variable.name))
  return(start.tab)
}
```

```
mean.starting.time(all.dat = backfilled.dat, variable.name = metformin.name,
    t1.name = t1.name, id.name = id.name, by = intervention.name)
```

```
   intervention Mean Days to Start metformin
1:            0                     33.37901
2:            1                     35.51101
```

```
mean.starting.time(all.dat = backfilled.dat, variable.name = statin.name, t1.name = t1.name,
    id.name = id.name, by = intervention.name)
```

```
   intervention Mean Days to Start statin
1:            0                   23.84413
2:            1                   24.56612
```

Rates of Adherence to Medications

- Now we want to track the rate of usage of each medication (with a filled prescription as a proxy for true adherence) over the first 6 months.
- Here we need to carefully consider the missing records. Not everyone will have a full 6 months of records!
- However, we can use the percentage of the recorded time that a prescription was filled as a reasonable estimate of each patient's rate of adherence.
- Then we will report the average of the rates across all of the patients in each cohort.

Calculating the Rates of Adherence by Cohort

```
calculate.adherence.rate <- function(x, t1, t2, max.time) {
    t2 <- pmin(t2, max.time)
    the.rate <- sum(x * (t2 - t1), na.rm = TRUE)/sum(t2 - t1, na.rm = TRUE)
    return(the.rate)
}

the.threshold <- one.year/2
the.rates <- dat[get(t1.name) < the.threshold, lapply(X = .SD, FUN = "calculate.adherence.rate",
    t1 = get(t1.name), t2 = get(t2.name), max.time = the.threshold), .SDcols = meds,
    by = c(id.name, intervention.name)]

mean.rates <- the.rates[, lapply(X = .SD, FUN = "mean", na.rm = TRUE), .SDcols = meds,
    by = intervention.name]
datatable(data = mean.rates[, lapply(X = .SD, FUN = "round.numerics", digits = 3)],
    rownames = FALSE)
```

Show 10 entries Search:

intervention	metformin	statin
0	0.521	0.616
1	0.511	0.612

Showing 1 to 2 of 2 entries Previous 1 Next

Hospitalizations

- **An important outcome**: hospitalizations are signs of deteriorating health and major costs.
- Reducing the rates of hospitalizations is an important goal – and one of the major reasons why the intervention was planned in the first place.
- Multiple questions to address.

Quantities to Investigate

- What percentage of patients in each cohort had at least one hospitalization in the first 6 months?
- Looking at the first 6 months, what is the average number of days spent in the hospital per person per year in each cohort?
- How many distinct hospitalizations were there?
- What is the relationship between the baseline rates of the other variables and whether a patient is hospitalized within the first 6 months?

Percentage with a Hospitalization within 6 Months

```
hospital.name <- "hospital"
hospitalized.within.6 <- dat[t1 < one.year/2, .(V1 = max(get(hospital.name),
    na.rm = TRUE)), by = c(id.name, intervention.name)]
setnames(x = hospitalized.within.6, old = "V1", new = hospital.name)
hospital.rates.by.cohort <- hospitalized.within.6[, .(`Percentage Hospitalized Within 6 Months` = 100 *
    mean(get(hospital.name), na.rm = TRUE)), by = intervention.name]
datatable(data = hospital.rates.by.cohort[, lapply(X = .SD, FUN = "round.numerics",
    digits = 1)], rownames = FALSE)
```

Show 10 entries Search:

intervention	Percentage Hospitalized Within 6 Months
0	57.3
1	53.6

Showing 1 to 2 of 2 entries Previous 1 Next

Average Number of Days in the Hospital

- Let's create a table for the records in the first 6 months.

```
days.hospitalized.first6 <- dat[get(t1.name) < one.year/2,]
days.hospitalized.first6[, eval(t2.name) := pmin(one.year/2, get(t2.name))]
days.hospitalized.first6[, days_in_period := (get(t2.name) - get(t1.name))]
days.hospitalized.first6[, days_in_hospital := days_in_period * get(hospital.name)]

days.tab <- days.hospitalized.first6[, .(`Number of Patients` = length(unique(get(id.name))), `Days Hospitalized` =
  sum(days_in_hospital, na.rm = TRUE), `Days of Follow-Up` = sum(days_in_period, na.rm = TRUE)), by =
  intervention.name]
days.tab[, `Years of Follow-Up` := `Days of Follow-Up`/one.year]
days.tab[, `Days Hospitalized Per Patient Per Year` := `Days Hospitalized` / `Years of Follow-Up`]
datatable(data = days.tab[, lapply(X = .SD, FUN = "round.numerics", digits = 1)], rownames = FALSE)
```

Show 10 entries Search:

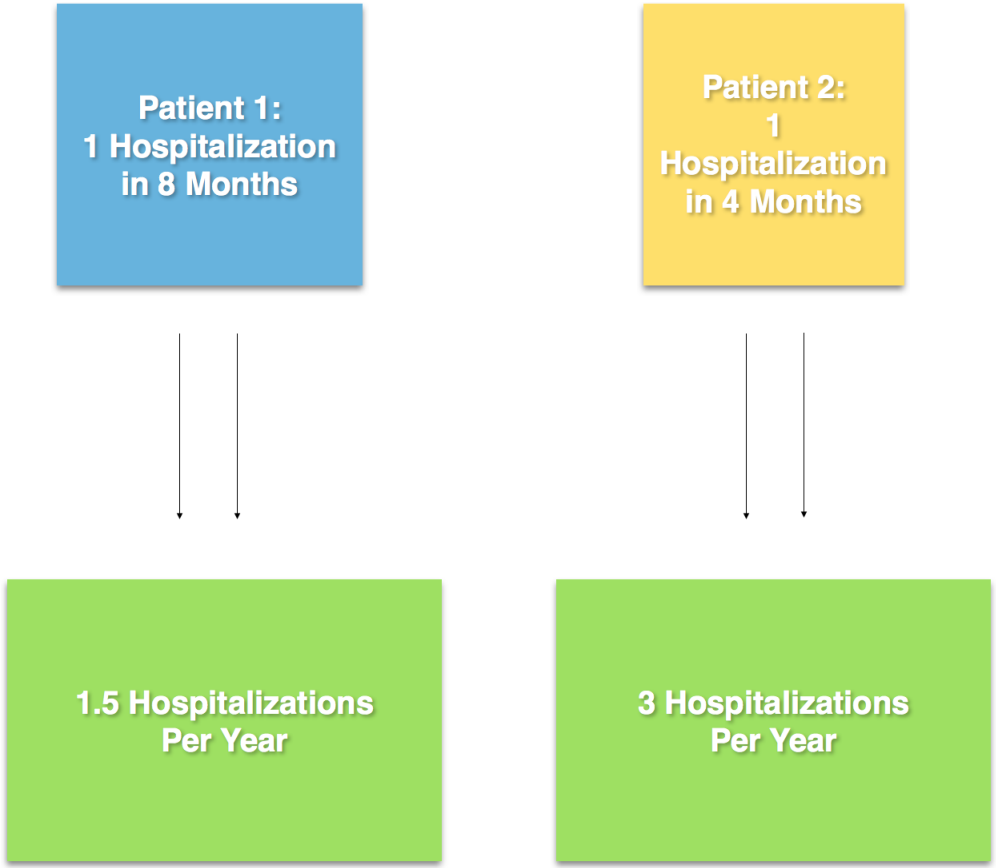
intervention	Number of Patients	Days Hospitalized	Days of Follow-Up	Years of Follow-Up	Days Hospitalized Per Patient Per Year
0	5014	21982.2	876716.8	2400.3	9.2
1	4986	19828	871394.1	2385.7	8.3

Showing 1 to 2 of 2 entries Previous 1 Next

Communication Alert

- Understanding the concept of the **average number of events per patient per year** is quite challenging.
- It's meant to say that all of the periods of follow-up can be treated equally. Then you can count the overall number of events relative to the years of follow-up in each group. Standardizing this to a number of events per person per year enables a fair comparison.
- But that's not easy to keep straight. So let's explain it better to our colleagues.

The Story of Two Patients



- Each patient had one hospitalization.
- However, one patient was followed for twice as long, 8 months versus 4.

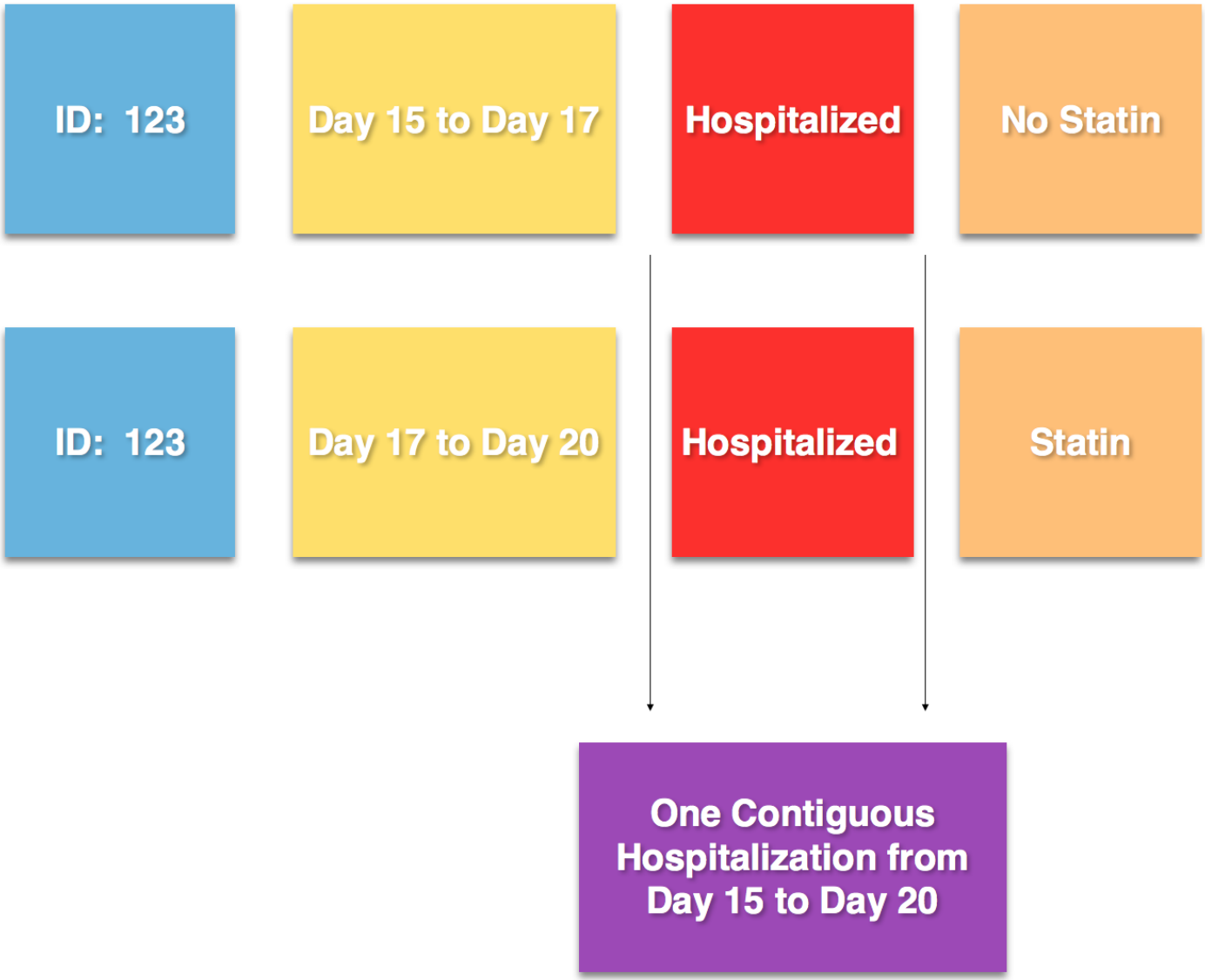
- In annualized terms, the event rates depend on the overall amount of follow-up.

Overall Number of Hospitalizations

- Each hospitalization covers a distinct period of time from when the patient is **admitted** until the patient is **discharged**.
- The overall number of hospitalizations is also of interest, even apart from the total number of days, because each admission is an adverse outcome that requires a major intervention.
- However, counting hospitalizations in panel data can be difficult…

Contiguity of a Hospitalization

- One contiguous hospitalization can span multiple records when the **other variables change** while the patient remains in the hospital.


ID: 123
Day 15 to Day 17
Hospitalized
No Statin
ID: 123
Day 17 to Day 20
Hospitalized
Statin
One Contiguous Hospitalization from Day 15 to Day 20

Can We Resolve This?

- Yes, but I'm going to make you do it yourself on the homework!

Linking Baseline Factors to Outcomes

Now we can start to address more complex questions:

- What can we say about the intervention's ability to **keep patients out of the hospital** for at least 6 months?
- What is the correlation between some of the other baseline factors and hospitalizations at 6 months?
- And what else should we be measuring?

The Intervention's Effect

- We previously saw the following charts:

```
datatable(data = hospital.rates.by.cohort[, lapply(X = .SD, FUN = "round.numerics",
    digits = 1)], rownames = FALSE)
```

Show 10 entries Search:

intervention	Percentage Hospitalized Within 6 Months
0	57.3
1	53.6

Showing 1 to 2 of 2 entries Previous 1 Next

```
datatable(data = days.tab[, lapply(X = .SD, FUN = "round.numerics", digits = 1)],
    rownames = FALSE)
```

Show 10 entries Search:

intervention	Number of Patients	Days Hospitalized	Days of Follow-Up	Years of Follow-Up	Days Hospitalized Per Patient Per Year

intervention	Number of Patients	Days Hospitalized	Days of Follow-Up	Years of Follow-Up	Days Hospitalized Per Patient Per Year
0	5014	21982.2	876716.8	2400.3	9.2
1	4986	19828	871394.1	2385.7	8.3

Showing 1 to 2 of 2 entries

Previous 1 Next

Interesting Evidence

- The results appears to show that the intervention leads to some improvements relative to the control.
- These results could be further substantiated – e.g. with statistical tests of significance and confidence intervals.
- Given that the data arose from a randomized, controlled clinical trial, a significant effect would likely indicate that the intervention is making substantial improvements in the health of the patients.

Linking Outcomes to Baseline Factors

- Let's construct an overall table showing the baseline factors along with an indicator of whether the patients were hospitalized within 6 months.

```
outcomes.dat <- hospitalized.within.6[, .(V1 = get(id.name), V2 = get(hospital.name))]
hosp6.name <- sprintf("%s_6", hospital.name)
setnames(x = outcomes.dat, old = c("V1", "V2"), new = c(id.name, hosp6.name))
bdat <- merge(x = baseline, y = outcomes.dat, by = id.name)
```

Correlations

```
the.factors <- c(intervention.name, age.name, weight.name, a1c.name, metformin.name,
    statin.name)
the.correlations <- bdat[, lapply(X = .SD, FUN = "cor", y = get(hosp6.name),
    use = "pairwise.complete.obs"), .SDcols = the.factors]
datatable(data = the.correlations[, lapply(X = .SD, FUN = "round.numerics",
    digits = 3)], rownames = FALSE)
```

Show 10 entries Search:

intervention	age	weight	a1c	metformin	statin
-0.038	0.301	0.048	0.07	-0.023	-0.024

Showing 1 to 1 of 1 entries Previous 1 Next

- For the baseline factors, only the patient's age has even a moderate correlation with hospitalizations.
- However, even small correlations can have an impact when we're talking about a large number of patients. Most of these variables do have a significant impact on the risk of hospitalizations in these data. But that's a lesson for another day…

And On and On

We are only scratching the surface of what you can investigate with panel data. You could ask many other questions, like:

- How long does a patient need to stay on a medicine?
- What kind of reductions in weight and A1C scores are needed to create meaningful reductions in a patient's risk?
- Are there factors that explain why a patient might be likely to be **readmitted** to the hospital within 30 days of a previous discharge?

All of these questions – and many others – would be quite interesting to explore. If you find a suitable data set, they could form the basis of an interesting topic for your final project.