Math 6040/7260 Linear Models

Mon/Wed/Fri 1:00pm - 1:50pm

Instructor: Dr. Xiang Ji, xji4@tulane.edu

1 Lecture 1:Jan 13

Today

- Introduction
- Introduce yourself
- Course logistics

What is this course about?

The term "linear models" describes a wide class of methods for the statistical analysis of multivariate data. The underlying theory is grounded in linear algebra and multivariate statistics, but applications range from biological research to public policy. The objective of this course is to provide a solid introduction to both the theory and practice of linear models, combining mathematical concepts with realistic examples.

Prerequisite

- **Must:** Introduction to Probability (Math 3070/6070), Mathematical Statistics (Math 3080/6080)
- **Good to have:** Scientific Computation II

A hierarchy of linear models

- The linear mean model:

$$\underset{n\times 1}{\mathbf{y}} = \underset{n\times p}{\mathbf{X}}\underset{p\times 1}{\boldsymbol{\beta}} + \underset{n\times 1}{\boldsymbol{\epsilon}}$$

  where $\mathbf{E}(\boldsymbol{\epsilon}) = \mathbf{0}$. Only assumption is that errors have mean 0.

- Gauss-Markov model:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

  where $\mathbf{E}(\boldsymbol{\epsilon}) = \mathbf{0}$ and $\mathbf{Var}(\boldsymbol{\epsilon}) = \sigma^2\mathbf{I}$. Uncorrelated errors with constant variance.

- Aitken model or general linear model:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

  where $\mathbf{E}(\boldsymbol{\epsilon}) = \mathbf{0}$ and $\mathbf{Var}(\boldsymbol{\epsilon}) = \sigma^2\mathbf{V}$. $\mathbf{V}$ is fixed and known.

- Variance components models: $\mathbf{y} \sim N(\mathbf{X}\beta, \sigma_1^2\mathbf{V}_1+\sigma_2^2\mathbf{V}_2+\cdots+\sigma_r^2\mathbf{V}_r)$ with $\mathbf{V}_1, \mathbf{V}_2, \ldots, \mathbf{V}_r$ known.
- General mixed linear Model:
$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$
where $\mathbf{E}(\boldsymbol{\epsilon}) = \mathbf{0}$ and $\mathbf{Var}(\boldsymbol{\epsilon}) = \boldsymbol{\Sigma}(\theta)$.
- Generalized linear models (GLMs). Logistic regression, probit regression, log-linear model (Poisson regression), ... Note the difference from the general linear model. GLMs are generalization of the *concept* of linear models. They are covered in Math 7360 - Data Analysis class (https://tulane-math-7360-2023.github.io/).

Syllabus

Check course website frequently for updates and announcements.

https://tulane-math-7260-2025.github.io/

HW submission

Through Github with demo on Friday class.

Presentations

Let me know your pick by the end of Friday (01/17/2025).

Spring 2023 comments

1. Experience in this course
    - I enjoyed this class a lot. I enjoyed that the class only had a few students, which made it feel like a community.
    - It was a bit hard to follow towards the end of the semester. I think we spent too much time on easier topics at the beginning of the semester but relatively less time on harder topics towards the end
    - The course was interesting, and the professor did a good job of thoroughly teaching the topics discussed.
2. Strong aspects of this course
    - Having the lectures based off of lecture notes was very nice because I could just listen to Dr. Ji and didn't need to worry about copying down a ton of material, which helped me learn better
    - I really enjoyed the setup of the course: prewritten notes for lectures, lab sessions for application and learning computing, presentations to learn fun material,

homework graded for completeness, take home exams. I feel like this setup helped me enjoy and learn the material without feeling stressed.

- The expectations were clearly presented, and the exams fairly reflected the materials we discussed in class. The content was also discussed thoroughly.

Spring 2022 comments

1. Experience in this course
   - I really enjoyed this class.
   - The lecture is dry. Expect Dr. Ji to read through the lecture notes until someone asks a question. Also, having a background in statistics gives context to the lecture notes.
     Response: I did emphasize on pre-requisite last year, but it still seemed not enough.
   - I think this course was well laid out. Even though I did take it as a graduation requirement, I ended up enjoying the course. I also think the work load in manageable and professor Xi does provide all the tools necessary to succeed in the course. It can be overwhelming at first but with a little time and effort you can get the hang of the material.
     Response: Mind the typos, students.
2. Strong aspects of this course
   - Professor Ji is a great professor.
   - Dr. Ji provides an inhuman amount of course material to help supplement learning, it was extremely helpful to have labs (answered and unanswered) as well as homework keys posted.
     Response: There will be labs again.
   - Notes are very structured and the professor is nice.
   - The strongest aspect of this course is the homework. They are a great way to interact with and learn the material. The problems can seem challenging, but are doable with a little effort. Also, he publishes the keys afterword so you can check your work and see what the reasoning behind the answer is. This coupled with the lab sessions are a great way to prepare for the exams.
3. There will be an internal mid-term-ish evaluation for this course. Will remember to go over them.

Spring 2021 comments

1. Experience in this course

- Overall, I had a pretty good experience in this course. It moved quickly, but that is expected from this level of course. Sometimes it was hard to stay engaged with the lectures and to really absorb the course material. Because the lectures moved so fast, I really appreciated how Professor made the full notes available at the time of the lecture. I would have liked if there were a few more examples with the notes, as sometimes the homework felt disjoint from the notes.
  Response: I will try to move slower this semester. I will start lab sessions earlier too.

- The professor is an extremely intelligent, kind, and understanding professor. He prioritizes in making sure that we understand the material and seeing how the material can be applied. His lecture notes were a godsend because the texts could be a bit ambiguous at times but he elucidated the material in such a comprehensible manner.
  Response: I will try to fix the left-over typos.

- Mentioned in class from other students/internal evaluation, conveying the mathematical concepts through the presentation is not a good idea to follow the class in real-time. Prepared presentation can give rise to a distraction on what we have been going over.
  Response: I am still delivering this class in hybrid-mode. I found the presentations fit online teaching better. I think the difficulty might be caused by (1) fast moving lecture (2) I only realized the need of reviewing basic concepts of probability almost a quarter into the semester...

- I found the setup of the course not very engaging. Additionally, many of the class notes came directly from the additional sources with no additional information or explanation, which I found to be not very helpful.
  Response: I actually like them. I was the guinea pig to test them.

- Easily help us to understand the main course, and the notes and details are great.
  Response: There will be notes.

- Moves very quickly and can be hard to keep up with. Sometimes instructions are unclear.
  Response: I will try to slow down.

- Both the instructor and the TA were helpful. It was hard to follow along in class though.
  Response: We don't have TA this time. Make use of the office hour. And I have to say, it needs effort to ace in this class.

2. Strong aspects of this course

   - Having the lecture notes and labs available was very helpful. Professor was also always very nice and accommodating, and willing to meet with me when I needed help. He also always responded to student feedback, if we asked for an extra day or two on the homework or something like that.
     Response: Here is an example of correctly using the office hours.

- His lecture notes and the lab sessions.
  Response: They will be there again.
- Lab session is necessarily required to this class. A lot of computations in the class would be done by computer due to the complexity, and students are expected to handle with the computer programming properly at a desired level. The course can be an introduction to the statistical computation, which does not exist in the mathematics department.
  Response: Hmm, there is a course Math 7360 Data Analysis that focuses more on the computational side.
- I appreciated the homework reviews in class and felt these helped clarify the material.
  Response: Of course, the reviews will be there again. The purpose of the course is for you to learn.
- Grading was easy which made up for the rigor.
  Response: Don't rely on this...
- Really appreciate that Professor Xiang made such a neat and tidy notes for us. It is really helpful for me to review. And notes have a great interaction with us, Professor Xiang also leaves some questions to help us think about the logic behind.
  Response: Well, Xiang is my first name. Please call me Prof. X.
- Prof. Xiang was highly organized and wanted his students to understand the course content more than he made them worry about grades. I learned a lot about Linear Models and feel confident applying the course content professionally and academically. I wish most of the Math department had his teaching style and implemented his course documents and organization structure. Prof. Xiang made the course content in class digestible and if I needed to review the material I could easily find it through his course notes and textbook. I wish I could say the same about my other courses.
  Response: Hmm, I like Prof. X. better.
- I really appreciated the emphasis on learning. It allowed for most students to take it at the pace that was good for them.
  Response: Please don't let your score rely on this comment.

3. There will be an internal mid-term-ish evaluation for this course. Will remember to go over them.

Rate my professor comments

N.A.

2 Lecture 2:Jan 15

Last time

- Introduction
- Course logistics

Today

- Reply to the "Presentation Dates" thread on Canvas by the end of Friday.
- Git

What is git?

Git is currently the most popular system for version control according to Google Trend. Git was initially designed and developed by Linus Torvalds in 2005 for Linux kernel development. Git is the British English slang for unpleasant person.

Why using git?

- GitHub is becoming a de facto central repository for open source development.
- **Advertise** yourself through GitHub (e.g., host a free personal webpage on GitHub, example and tutorial).
- a skill that employers look for (according to this AmStat article).

Git workflow

Figure 2.1 shows its basic workflow.

What do I need to use Git?

- A **Git server** enabling multi-person collaboration through a centralized repository.
- A **Git client** on your own machine.
  - Linux: Git client program is shipped with many Linux distributions, e.g., Ubuntu and CentOS. If not, install using a package manager, e.g., `yum install git` on CentOS.
  - Mac: follow instructions at https://www.atlassian.com/git/tutorials/install-git.
  - Windows: Git for Windows at https://gitforwindows.org (Graphical User Interface, or in short, GUI) aka Git Bash.
- Do **not** totally rely on GUI or IDE (Integrated Development Environment). Learn to use Git on command line, which is needed for cluster and cloud computing.

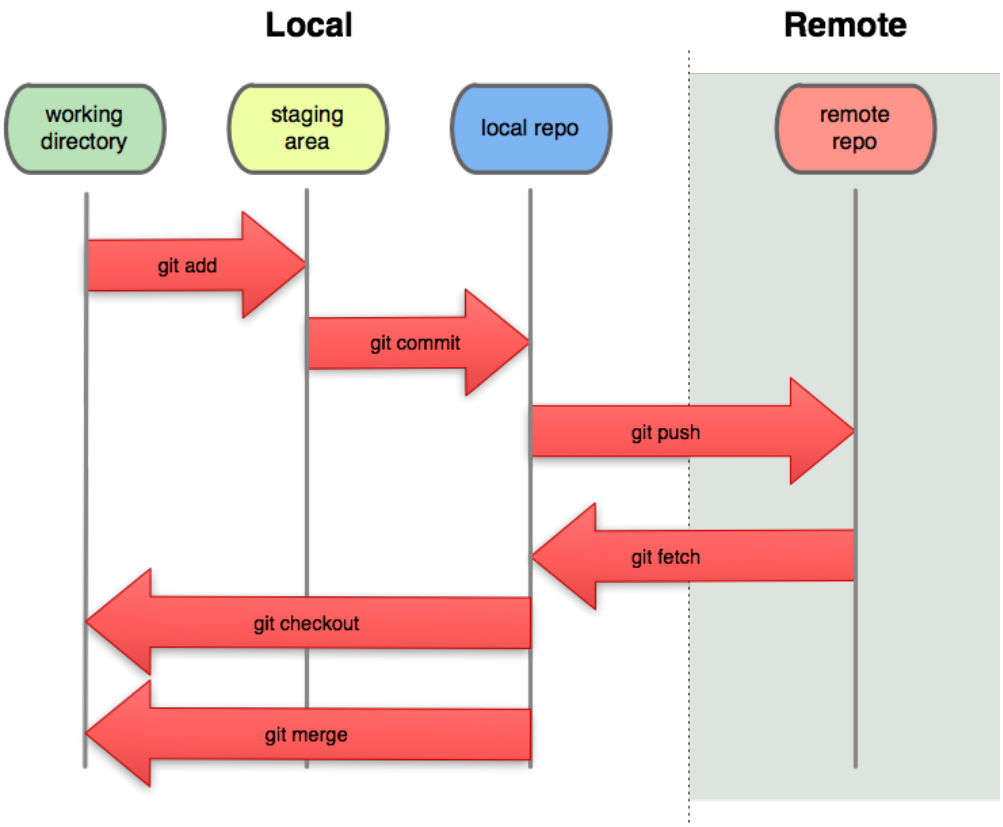

Figure 2.1

Git survival commands

- `git pull` synchronize local Git directory with remote repository.
- Modify files in local working directory.
- `git add FILES` add snapshots to staging area
- `git commit -m "message"` store snapshots permanently to (**local**) Git repository
- `git push` push commits to remote repository.

Git basic usage

Working with your local copy.

- `git pull` : update local Git repository with remote repository (fetch + merge).
- `git log FILENAME` : display the current status of working directory.
- `git diff` : show differences (by default difference from the most recent commit).
- `git add file1 file2 ...` : add file(s) to the staging area.

- `git commit` : commit changes in staging area to Git directory.
- `git push` : publish commits in local Git repository to remote repository.
- `git reset –soft HEAD 1` : undo the last commit.
- `git checkout FILENAME` : go back to the last commit, discarding all changes made.
- `git rm FILENAME` : remove files from git control.

Git demonstration

Show how to create a private git repository for HW and Exam submissions.

On GitHub

- Obtain student developer pack.
- Create a private repository `Xiang-Ji-math-6040-2025-spring` (please substitute 6040 by 7260 if you are taking the graduate level and use your own first and last names). Add `xji3` as your collaborators with write permission (instruction).

On your local machine:

- clone the repository: please refer to this webpage with instructions for your operating system.
- enter the folder: `cd Xiang-Ji-math-6040-2025-spring` .
- after finishing the rest of the questions, save your file inside your git repository folder `Xiang-Ji-math-6040-2025-spring` with name `hw1.pdf` (for example). Please make it human-readable.
- now using git commands to stage this change: `git add hw1.pdf`
- commit: `git commit -m “hw1 submission”` (remember to replace the quotation mark)
- push to remote server: `git push`
- tag version hw1: `git tag hw1` and push: `git push --tags` .

Take a look at the tags on GitHub (instructions).

When submitting your hw, please email your instructor (xji4@tulane.edu) a link to your tag (instructions).

3 Lecture 3:Jan 17

Last time

- Git

Today

- HW1 posted
- Linear algebra: vector and vector space, rank of a matrix
- Column space and Nullspace (JM Appendix A)

Notations

$$\underset{n\times 1}{\mathbf{y}} = \underset{n\times p}{\mathbf{X}}\underset{p\times 1}{\boldsymbol{\beta}} + \underset{n\times 1}{\boldsymbol{\epsilon}}$$

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{pmatrix}$$

- All vectors are column vector
- Write dimensions underneath as in $\underset{n\times p}{\mathbf{X}}$ or as $\mathbf{X} \in \mathbb{R}^{n\times p}$
- Bold upper-case letters for Matrices. Bold lower-case letters for Vectors.

Vector and vector space

(from JM Appendix A)

- A set of vectors $\mathbf{x}_1, \dots, \mathbf{x}_n$ are *linearly dependent* if there exist coefficients c_j for $j = 1, 2, \dots, n$ such that $\sum_{j=1}^n c_j \mathbf{x}_j = \mathbf{0}$ and $||\mathbf{c}||_2 = \sqrt{\sum_{j=1}^n c_j^2} > 0$. They are *linearly independent* if $\sum_{j=1}^n c_j \mathbf{x}_j = \mathbf{0}$ implies ($\Longrightarrow$) $c_j = 0$ for all j.
- Two vectors are *orthogonal* to each other, written $\mathbf{x} \perp \mathbf{y}$, if their inner product is 0, that is $\mathbf{x}^T\mathbf{y} = \mathbf{y}^T\mathbf{x} = \sum_j x_j y_j = 0$.
- A set of vectors $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(n)}$ are mutually orthogonal iff ($\Longleftrightarrow$) $\mathbf{x}^{(i)T}\mathbf{x}^{(j)} = 0$ for $\forall i \neq j$.
- The most common set of vectors that are mutually orthogonal are the *elementary* vectors $\mathbf{e}^{(1)}, \mathbf{e}^{(2)}, \dots, \mathbf{e}^{(n)}$, which are all zero, except for one element equal to 1, so that $\mathbf{e}_i^{(i)} = 1$ and $\mathbf{e}_j^{(i)} = 0, \forall j \neq i$.

- A *vector space* $\mathcal{S}$ is a set of vectors that are closed under addition and scalar multiplication, that is
  - if $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ are in $\mathcal{S}$, then $c_1\mathbf{x}^{(1)} + c_2\mathbf{x}^{(2)}$ is in $\mathcal{S}$.
- A vector space $\mathcal{S}$ is *generated* or *spanned* by a set of vectors $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}$, written as $\mathcal{S} = \text{span}\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}\}$, if any vector $\mathbf{x}$ in the vector space is a linear combination of $\mathbf{x}_i, i = 1, 2, \ldots, n$.
- A set of linearly independent vectors that generate or span a space $\mathcal{S}$ is called a *basis* of $\mathcal{S}$.

Example A.1

Let

$$\mathbf{x}^{(1)} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \mathbf{x}^{(2)} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}, \text{ and } \mathbf{x}^{(3)} = \begin{bmatrix} -3 \\ -1 \\ 1 \\ 3 \end{bmatrix}.$$

Then $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ are linearly independent, but $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$, and $\mathbf{x}^{(3)}$ are linearly dependent since $5\mathbf{x}^{(1)} - 2\mathbf{x}^{(2)} + \mathbf{x}^{(3)} = 0$

Rank

Some matrix concepts arise from viewing columns or rows of the matrix as vectors. Assume $\mathbf{A} \in \mathbb{R}^{m \times n}$.

- $\text{rank}(\mathbf{A})$ is the maximum number of linearly independent rows or columns of a matrix.
- $\text{rank}(\mathbf{A}) \leqslant \min\{m, n\}$.
- A matrix is *full* rank if $\text{rank}(\mathbf{A}) = \min\{m, n\}$. It is *full row rank* if $\text{rank}(\mathbf{A}) = m$. It is *full column rank* if $\text{rank}(\mathbf{A}) = n$.
- a square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is *singular* if $\text{rank}(\mathbf{A}) < n$ and *non-singular* if $\text{rank}(\mathbf{A}) = n$.
- $\text{rank}(\mathbf{A}) = \text{rank}(\mathbf{A}^T) = \text{rank}(\mathbf{A}^T\mathbf{A}) = \text{rank}(\mathbf{A}\mathbf{A}^T)$. (Show this in HW.)
- $\text{rank}(\mathbf{AB}) \leqslant \min\{\text{rank}(\mathbf{A}), \text{rank}(\mathbf{B})\}$. (Hint: Columns of $\mathbf{AB}$ are spanned by columns of $\mathbf{A}$ and rows of of $\mathbf{AB}$ are spanned by rows of $\mathbf{B}$.)
- if $\mathbf{Ax} = \mathbf{0}_m$ for some $\mathbf{x} \neq \mathbf{0}_n$, then $\text{rank}(\mathbf{A}) \leqslant n - 1$.

Column space

Definition: The column space of a matrix, denoted by $\mathcal{C}(\mathbf{A})$ is the vector space spanned by the columns of the matrix, that is,

$$\mathcal{C}(\mathbf{A}) = \{\mathbf{x} : \text{ there exists a vector } \mathbf{c} \text{ such that } \mathbf{x} = \mathbf{Ac}\}.$$

This means that if $\mathbf{x} \in \mathcal{C}(\mathbf{A})$, we can find coefficients c_j such that

$$\mathbf{x} = \sum_j c_j \mathbf{a}^{(j)}$$

where $\mathbf{a}^{(j)} = \mathbf{A}_{\cdot j}$ denotes the jth column of matrix $\mathbf{A}$.

- The column space of a matrix consists of all vectors formed by multiplying that matrix by any vector.
- The number of basis vectors for $\mathcal{C}(\mathbf{A})$ is then the number of linearly independent columns of the matrix $\mathbf{A}$, and so, $\dim(\mathcal{C}(\mathbf{A})) = \text{rank}(\mathbf{A})$.
- The dimension of a space is the number of vectors in its basis.

Example A.2

Let $\mathbf{A} = \begin{bmatrix} 1 & 1 & -3 \\ 1 & 2 & -1 \\ 1 & 3 & 1 \\ 1 & 4 & 3 \end{bmatrix}$ and $\mathbf{c} = \begin{bmatrix} 5 \\ 4 \\ 3 \end{bmatrix}$. Show that $\mathbf{Ac}$ is a linear combination of columns in $\mathbf{A}$.

solution:

$$\mathbf{Ac} = \begin{bmatrix} 1 \times 5 + 1 \times 4 + (-3) \times 3 \\ 1 \times 5 + 2 \times 4 + (-1) \times 3 \\ 1 \times 5 + 3 \times 4 + 1 \times 3 \\ 1 \times 5 + 4 \times 4 + 3 \times 3 \end{bmatrix} = \begin{bmatrix} 0 \\ 10 \\ 20 \\ 30 \end{bmatrix}.$$

You could recognize that

$$\mathbf{Ac} = 5 \times \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} + 4 \times \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} + 3 \times \begin{bmatrix} -3 \\ -1 \\ 1 \\ 3 \end{bmatrix} = 5\mathbf{a}^{(1)} + 4\mathbf{a}^{(2)} + 3\mathbf{a}^{(3)} = \begin{bmatrix} 0 \\ 10 \\ 20 \\ 30 \end{bmatrix}.$$

Result A.1

$\text{rank}(\mathbf{AB}) \leqslant \min(\text{rank}(\mathbf{A}), \text{rank}(\mathbf{B}))$.

proof: Each column of $\mathbf{AB}$ is a linear combination of columns of $\mathbf{A}$ (i.e. $(\mathbf{AB})_{\cdot j} = \mathbf{Ab}^{(j)}$), so the number of linearly independent columns of $\mathbf{AB}$ cannot be greater than that of A. Similarly, $\text{rank}(\mathbf{AB}) = \text{rank}(\mathbf{B}^T\mathbf{A}^T)$, the same argument gives $\text{rank}(\mathbf{B}^T)$ as an upper bound.

Result A.2

- (a) If $\mathbf{A} = \mathbf{BC}$, then $\mathcal{C}(\mathbf{A}) \subseteq \mathcal{C}(\mathbf{B})$.
- (b) If $\mathcal{C}(\mathbf{A}) \subseteq \mathcal{C}(\mathbf{B})$, then there exists a matrix $\mathbf{C}$ such that $\mathbf{A} = \mathbf{BC}$.

proof: For (a), any vector $\mathbf{x} \in \mathcal{C}(\mathbf{A})$ can be written as $\mathbf{x} = \mathbf{Ad} = \mathbf{B}(\mathbf{Cd})$.
For (b), $\mathbf{A}_{\cdot j} \in \mathcal{C}(B)$, so that there exists a vector $\mathbf{c}^{(j)}$ such that $\mathbf{A}_{\cdot j} = \mathbf{Bc}^{(j)}$. The matrix $\mathbf{C} = (\mathbf{c}^{(1)}, \mathbf{c}^{(2)}, \ldots, \mathbf{c}^{(n)})$ satisfies that $\mathbf{A} = \mathbf{BC}$.

4 Lecture 4: Jan 22

Last time

- Linear algebra: vector and vector space, rank of a matrix, Column space (JM Appendix A)

Today

- Nullspace
- Intro to R

Null space

Definition: The null space of a matrix, denoted by $\mathcal{N}(\mathbf{A})$, is $\mathcal{N}(\mathbf{A}) = \{\mathbf{y} : \mathbf{A}\mathbf{y} = \mathbf{0}\}$.

Result A.3

If $\mathbf{A}$ has full-column rank, then $\mathcal{N}(\mathbf{A}) = \{\mathbf{0}\}$.

proof: Matrix $\mathbf{A}$ has full-column rank means its columns are linearly independent, which means that $\mathbf{A}\mathbf{c} = \mathbf{0}$ implies $\mathbf{c} = \mathbf{0}$.

Theorem A.1

Assume $\mathbf{A} \in \mathbb{R}^{m\times n}$, then $\dim(\mathcal{C}(\mathbf{A})) = r$ and $\dim(\mathcal{N}(\mathbf{A})) = n - r$, where $r = \text{rank}(\mathbf{A})$.

See JM Appendix Theorem A.1 for the proof.
proof: Denote $\dim(\mathcal{N}(\mathbf{A}))$ by k, to be determined, and construct a set of basis vectors for $\mathcal{N}(\mathbf{A})$: $\left\{\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \ldots, \mathbf{u}^{(k)}\right\}$, so that $\mathbf{A}\mathbf{u}^{(i)} = \mathbf{0}$, for $i = 1, 2, \ldots, k$. Now, construct a basis for $\mathbb{R}^n$ by adding the vectors $\left\{\mathbf{u}^{(k+1)}, \ldots, \mathbf{u}^{(n)}\right\}$, which are not in $\mathcal{N}(\mathbf{A})$. Clearly, $\mathbf{A}\mathbf{u}^{(i)} \in \mathcal{C}(\mathbf{A})$ for $i = k+1, \ldots, n$, and so the span of these vectors form a subspace of $\mathcal{C}(\mathbf{A})$. These vectors $\left\{\mathbf{A}\mathbf{u}^{(i)}, i = k+1, \ldots, n\right\}$ are also linearly independent from the following argument: suppose $\sum_{i=k+1}^{n} c_i \mathbf{A}\mathbf{u}^{(i)} = \mathbf{0}$; then $\sum_{i=k+1}^{n} c_i \mathbf{A}\mathbf{u}^{(i)} = \mathbf{A}\left[\sum_{i=k+1}^{n} c_i \mathbf{u}^{(i)}\right] = \mathbf{0}$, and hence $\sum_{i=k+1}^{n} c_i \mathbf{u}^{(i)}$ is a vector in $\mathcal{N}(\mathbf{A})$. Therefore, there exist b_i such that $\sum_{i=k+1}^{n} c_i \mathbf{u}^{(i)} = \sum_{i=1}^{k} b_i \mathbf{u}^{(i)}$, or $\sum_{i=1}^{k} b_i \mathbf{u}^{(i)} - \sum_{i=k+1}^{n} c_i \mathbf{u}^{(i)} = \mathbf{0}$. Since $\left\{\mathbf{u}^{(i)}\right\}$ form a basis for $\mathbb{R}^n$, c_i must all be zero. Therefore $\mathbf{A}\mathbf{u}^{(i)}, i = k+1, \ldots, n$ are linearly independent. At this point, since $\text{span}\{\mathbf{A}\mathbf{u}^{(k+1)}, \ldots, \mathbf{A}\mathbf{u}^{(n)}\} \subseteq \mathcal{C}(\mathbf{A})$, $\dim(\mathcal{C}(\mathbf{A}))$ is at least $n - k$. Suppose there is a vector $\mathbf{y}$ that is in $\mathcal{C}(\mathbf{A})$, but not in the span; then there exists $\mathbf{u}^{(n+1)}$ so that $\mathbf{y} = \mathbf{A}\mathbf{u}^{(n+1)}$ and $\mathbf{u}^{(n+1)}$ is linearly independent of $\{\mathbf{u}^{(k+1)}, \ldots, \mathbf{u}^{(n)}\}$ (and clearly not in $\mathcal{N}(\mathbf{A})$), making $n+1$ linearly independent vectors in $\mathbb{R}^n$. Since that is not possible, the span is equal to $\mathcal{C}(\mathbf{A})$ and $\dim(\mathcal{C}(\mathbf{A})) = n - k = r = \text{rank}(\mathbf{A})$, so that $k = \dim(\mathcal{N}(\mathbf{A})) = n - r$.

Interpretation: "dimension of column space + dimension of null space = # columns"
Mis-Interpretation: Columns space and null space are orthogonal complement to each other. They are of different orders in general!

Lecture 5, R Basics

MATH-7260 Linear Models

Dr. Xiang Ji @ Tulane University

January 27, 2025

Contents

Announcement

- Just stay warm.

R basics

styles

(reading assignment)

Checkout Style guide in Advanced R and the tidyverse style guide.

Arithmetic

R can do any basic mathematical computations.

symbol	use
+	addition
-	subtraction
*	multiplication
/	division
^	power

symbol	use
%%	modulus
exp()	exponent
log()	natural logarithm
sqrt()	square root
round()	rounding
floor()	flooring
ceiling()	ceiling

Objects

You can create an R object to save results of a computation or other command.

Example 1

```
x <- 3 + 5
x
```

```
## [1] 8
```

- In most languages, the direction of passing through the value into the object goes from right to left (e.g. with "="). However, R allows both directions (which is actually bad!). In this course, we encourage the use of "<-" or "=". There are people liking "=" over "<-" for the reason that "<-" sometimes break into two operators "< -".

Example 2

```
x < - 3 + 5
```

```
## [1] FALSE
```

```
x
```

```
## [1] 8
```

- For naming conventions, stick with either "." or "_" (refer to the style guide).

Example 3

```
sum.result <- x + 5
sum.result
```

```
## [1] 13
```

- *important*: many names are already taken for built-in R functions. Make sure that you don't override them.

Example 4

```
sum(2:5)
```

```
## [1] 14
```

```
sum
```

```
## function (..., na.rm = FALSE)  .Primitive("sum")
```

```
sum <- 3 + 4 + 5
sum(5:8)
```

```
## [1] 26
```

```
sum
```

```
## [1] 12
```

- R is case-sensitive. “Math.7260” is different from “math.7260”.

Locating and deleting objects:

The commands “objects()” and “ls()” will provide a list of every object that you’ve created in a session.

```
objects()
```

```
## [1] "sum"        "sum.result" "x"
```

```
ls()
```

```
## [1] "sum"        "sum.result" "x"
```

The “rm()” and “remove()” commands let you delete objects (tip: always clearn-up your workspace as the first command)

```
rm(list=ls())  # clean up workspace
```

Vectors

Many commands in R generate a vector of output, rather than a single number.

The “c()” command: creates a vector containing a list of specific elements.

Example 1

```
c(7, 3, 6, 0)
```

```
## [1] 7 3 6 0
```

```
c(73:60)
```

```
##  [1] 73 72 71 70 69 68 67 66 65 64 63 62 61 60
```

```
c(7:3, 6:0)
```

```
##  [1] 7 6 5 4 3 6 5 4 3 2 1 0
```

```
c(rep(7:3, 6), 0)
```

```
##  [1] 7 6 5 4 3 7 6 5 4 3 7 6 5 4 3 7 6 5 4 3 7 6 5 4 3 0
```

Example 2 The command “seq()” creates a sequence of numbers.

```
seq(7)
```

```
## [1] 1 2 3 4 5 6 7
```

```
seq(3, 70, by = 6)
```

```
##  [1]  3  9 15 21 27 33 39 45 51 57 63 69
```

```
seq(3, 70, length = 6)
```

```
## [1]  3.0 16.4 29.8 43.2 56.6 70.0
```

Operations on vectors

Use brackets to select element of a vector.

```
x <- 73:60
x[2]
```

```
## [1] 72
```

```r
x[2:5]
```

```
## [1] 72 71 70 69
```

```r
x[-(2:5)]
```

```
##  [1] 73 68 67 66 65 64 63 62 61 60
```

Can access by "name" (safe with column/row order changes)

```r
y <- 1:3
names(y) <- c("do", "re", "mi")
y[3]
```

```
## mi
##  3
```

```r
y["mi"]
```

```
## mi
##  3
```

R commands on vectors

command	usage
sum()	sum over elements in vector
mean()	compute average value
sort()	sort elements in a vector
min(), max()	min and max values of a vector
length()	length of a vector
summary()	returns the min, Q1, median, mean, Q3, and max values of a vector
sample(x, size, replace = FALSE, prob = NULL)	takes a random sample from a vector with or without replacement

Exercise Write a command to generate a random permutation of the numbers between 1 and 5 and save it to an object.

Matrix

matrix() command creates a matrix from the given set of values

```r
matrix.example <- matrix(rnorm(100), nrow = 10, ncol = 10, byrow = TRUE)
matrix.example
```

```
##             [,1]       [,2]       [,3]        [,4]       [,5]        [,6]
##  [1,]  0.3666982 -1.7669433  1.7641834 -0.61621887  1.4863224 -0.94479077
##  [2,] -0.9857926 -1.2483619 -0.1926700  0.15690203  0.5847165 -1.93870114
##  [3,]  1.8374926 -0.1019404  0.3921290  1.66280129 -0.1326306  1.02612476
##  [4,] -0.1215171  1.7519705  1.7286263  0.56500359  0.0752892  2.00487245
##  [5,]  0.3809785 -1.1764969 -1.3117362 -1.77008120  0.4225604  0.38768775
##  [6,] -0.8081790  0.3031515  0.2896139  1.29129656  1.7642979 -0.01015978
##  [7,] -0.8699963  1.0416742 -0.2101917 -1.21447486 -1.7583827  0.78481029
##  [8,]  0.3492486 -1.5457233  0.9106507  0.07285106 -1.4851710  0.09121639
##  [9,] -0.6137160  0.2900285  2.1979912 -1.38086049 -1.5609317  0.35626883
## [10,]  1.9593308 -2.1541732  0.1223442  1.71848814 -1.0178962 -1.81570882
##             [,7]        [,8]       [,9]       [,10]
```

```
##  [1,] -1.1364874  2.18514605  0.7735370 -0.03889042
##  [2,] -0.3107539  0.95673779 -0.2807409 -0.16059619
##  [3,] -0.1206051 -0.23887993  2.2026695 -0.79009467
##  [4,]  0.2567208 -0.44896466 -1.1802590  0.82425547
##  [5,]  1.1109844 -0.05011302 -0.2438188 -0.19231284
##  [6,] -0.2056786 -0.01680366 -1.7923752 -0.92163776
##  [7,] -0.4687571  0.80052581  1.8038154 -0.24952519
##  [8,]  0.1769619 -1.24786462  1.5204527 -1.98957849
##  [9,]  0.6492282  0.84311980 -0.6476299 -0.49847309
## [10,] -1.5319584  0.40464881  0.5350702 -0.17699616
```

R commands on vector/matrix

command	usage
sum()	sum over elements in vector/matrix
mean()	compute average value
sort()	sort all elements in a vector/matrix
min(), max()	min and max values of a vector/matrix
length()	length of a vector/matrix
summary()	returns the min, Q1, median, mean, Q3, and max values of a vector
dim()	dimension of a matrix
cbind()	combine a sequence of vector, matrix or data-frame arguments and combine by columns
rbind()	combine a sequence of vector, matrix or data-frame arguments and combine by rows
names()	get or set names of an object
colnames()	get or set column names of a matrix-like object
rownames()	get or set row names of a matrix-like object

```
sum(matrix.example)
```

```
## [1] 1.488254
```

```
mean(matrix.example)
```

```
## [1] 0.01488254
```

```
sort(matrix.example)
```

```
##   [1] -2.15417318 -1.98957849 -1.93870114 -1.81570882 -1.79237520 -1.77008120
##   [7] -1.76694327 -1.75838269 -1.56093167 -1.54572329 -1.53195838 -1.48517104
##  [13] -1.38086049 -1.31173623 -1.24836192 -1.24786462 -1.21447486 -1.18025905
##  [19] -1.17649695 -1.13648736 -1.01789623 -0.98579258 -0.94479077 -0.92163776
##  [25] -0.86999626 -0.80817903 -0.79009467 -0.64762991 -0.61621887 -0.61371604
##  [31] -0.49847309 -0.46875710 -0.44896466 -0.31075391 -0.28074086 -0.24952519
##  [37] -0.24381878 -0.23887993 -0.21019167 -0.20567863 -0.19267004 -0.19231284
##  [43] -0.17699616 -0.16059619 -0.13263062 -0.12151707 -0.12060510 -0.10194035
##  [49] -0.05011302 -0.03889042 -0.01680366 -0.01015978  0.07285106  0.07528920
##  [55]  0.09121639  0.12234418  0.15690203  0.17696188  0.25672075  0.28961388
##  [61]  0.29002852  0.30315153  0.34924861  0.35626883  0.36669820  0.38097853
##  [67]  0.38768775  0.39212899  0.40464881  0.42256040  0.53507022  0.56500359
```

```
##  [73]  0.58471649  0.64922817  0.77353698  0.78481029  0.80052581  0.82425547
##  [79]  0.84311980  0.91065069  0.95673779  1.02612476  1.04167419  1.11098445
##  [85]  1.29129656  1.48632237  1.52045266  1.66280129  1.71848814  1.72862626
##  [91]  1.75197050  1.76418343  1.76429791  1.80381544  1.83749262  1.95933084
##  [97]  2.00487245  2.18514605  2.19799120  2.20266949
```

```
summary(matrix.example)
```

```
##        V1                 V2                 V3                 V4
##  Min.   :-0.9858   Min.   :-2.1542   Min.   :-1.3117   Min.   :-1.77008
##  1st Qu.:-0.7596   1st Qu.:-1.4714   1st Qu.:-0.1139   1st Qu.:-1.06491
##  Median : 0.1139   Median :-0.6392   Median : 0.3409   Median : 0.11488
##  Mean   : 0.1495   Mean   :-0.4607   Mean   : 0.5691   Mean   : 0.04857
##  3rd Qu.: 0.3774   3rd Qu.: 0.2999   3rd Qu.: 1.5241   3rd Qu.: 1.10972
##  Max.   : 1.9593   Max.   : 1.7520   Max.   : 2.1980   Max.   : 1.71849
##        V5                  V6                   V7                 V8
##  Min.   :-1.75838   Min.   :-1.938701   Min.   :-1.5320   Min.   :-1.2479
##  1st Qu.:-1.36835   1st Qu.:-0.711133   1st Qu.:-0.4293   1st Qu.:-0.1917
##  Median :-0.02867   Median : 0.223743   Median :-0.1631   Median : 0.1939
##  Mean   :-0.16218   Mean   :-0.005838   Mean   :-0.1580   Mean   : 0.3188
##  3rd Qu.: 0.54418   3rd Qu.: 0.685530   3rd Qu.: 0.2368   3rd Qu.: 0.8325
##  Max.   : 1.76430   Max.   : 2.004872   Max.   : 1.1110   Max.   : 2.1851
##        V9                V10
##  Min.   :-1.7924   Min.   :-1.9896
##  1st Qu.:-0.5559   1st Qu.:-0.7172
##  Median : 0.1456   Median :-0.2209
##  Mean   : 0.2691   Mean   :-0.4194
##  3rd Qu.: 1.3337   3rd Qu.:-0.1647
##  Max.   : 2.2027   Max.   : 0.8243
```

Exercise Write a command to generate a random permutation of the numbers between 1 and 5 and save it to an object.

Comparison (logic operator)

symbol	use
!=	not equal
==	equal
>	greater
>=	greater or equal
<	smaller
<=	smaller or equal
is.na	is it "Not Available"/Missing
complete.cases	returns a logical vector specifying which observations/rows have no missing values
is.finite	if the value is finite
all	are all values in a logical vector true?
any	any value in a logical vector is true?

```
test.vec <- 73:68
test.vec
```

```
## [1] 73 72 71 70 69 68
```

```
test.vec < 70
```

```
## [1] FALSE FALSE FALSE FALSE  TRUE  TRUE
```

```
test.vec > 70
```

```
## [1]  TRUE  TRUE  TRUE FALSE FALSE FALSE
```

```
test.vec[3] <- NA
test.vec
```

```
## [1] 73 72 NA 70 69 68
```

```
is.na(test.vec)
```

```
## [1] FALSE FALSE  TRUE FALSE FALSE FALSE
```

```
complete.cases(test.vec)
```

```
## [1]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
```

```
all(is.na(test.vec))
```

```
## [1] FALSE
```

```
any(is.na(test.vec))
```

```
## [1] TRUE
```

Now let's do a test of accuracy for doubles in R. Recall that for *Double* precision, we get approximately $\log_{10}(2^{52}) \approx 16$ decimal point for precision.

```
test.exponent <- -(7:18)
10^test.exponent == 0
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
1 - 10^test.exponent == 1
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE
```

```
7360 - 10^test.exponent == 7360
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

```
73600 - 10^test.exponent == 73600
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

Other operators

%in%, match

```
test.vec
```

```
## [1] 73 72 NA 70 69 68
```

```
66 %in% test.vec
```

```
## [1] FALSE
```

```
match(66, test.vec, nomatch = 0)
```

```
## [1] 0
```

```r
70 %in% test.vec
```

```
## [1] TRUE
```

```r
match(70, test.vec, nomatch = 0)
```

```
## [1] 4
```

```r
match(70, test.vec, nomatch = 0) > 0 # the implementation of %in%
```

```
## [1] TRUE
```

Control flow

These are the basic control-flow constructs of the R language. They function in much the same way as control statements in any Algol-like (Algol short for "Algorithmic Language") language. They are all reserved words.

keyword	usage
if	**if**(*cond*) *expr*
if-else	**if**(*cond*) *cons.expr* **else** *alt.expr*
for	**for**(*var* **in** *seq*) *expr*
while	**while**(*cond*) *expr*
break	breaks out of a *for* loop
next	halts the processing of the current iteration and advances the looping index

Define a function

Read Function section from Advanced R by Hadley Wickham. We will visit functions in more details.

```r
DoNothing <- function() {
  return(invisible(NULL))
}
DoNothing()
```

In general, try to avoid using loops (vectorize your code) in R. If you have to loop, try using **for** loops first. Sometimes, **while** loops can be dangerous (however, a smart *compiler* should detect this).

```r
DoBadThing <- function() {
  result <- NULL
  while(TRUE) {
    result <- c(result, rnorm(100))
  }
  return(result)
}
# DoBadThing()
```

Install packages

You can install R packages from several places (reference):

- Comprehensive R Archive Network (CRAN)
  - Official R packages repository
  - Some levels of code checks (cross platform support, version control etc)
  - Most common place you will install packages

  - Pick a mirror location near you
  - `install.packages("packge_name")`
- GitHub
  - May get development version of a package
  - Almost zero level of code checks
  - Common place to develop a package before submitting to CRAN

```
install.packages("devtools")
library(devtools)
install_github("tidyverse/ggplot2")
```

Load packages

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
require(tidyverse)
```