

UCLA-ALIGNED COURSE FIELD NOTES

STATS 102A

Introduction to Computational Statistics with R

Reproducible R, visualization, numerical methods, simulation, randomization, and bootstrap computation

How these notes are written. I wrote each chapter the way I wish the material had been explained the first time: the real idea, the point where I usually get stuck, the simplest mental model that still stays honest, and a worked decision instead of a polished answer appearing from nowhere. Every chapter closes with practice and a fully written solution. If a sentence sounds impressive but does not help me solve or explain something, it does not belong here.

Daniel Mastick

Curriculum map, working notes, practice, and solutions

Course map

Week	Core thread	What should be solid by the end
1	R objects, vectorization, and reliable functions	Reconstruct the model, work an application, test its assumptions, and explain the result in ordinary language.
2	Tidy data, transformation, and reproducibility	Reconstruct the model, work an application, test its assumptions, and explain the result in ordinary language.
3	Statistical graphics as model checks	Reconstruct the model, work an application, test its assumptions, and explain the result in ordinary language.
4	Numerical root finding and optimization	Reconstruct the model, work an application, test its assumptions, and explain the result in ordinary language.
5	Random-number generation and simulation design	Reconstruct the model, work an application, test its assumptions, and explain the result in ordinary language.
6	Permutation and randomization tests	Reconstruct the model, work an application, test its assumptions, and explain the result in ordinary language.
7	Bootstrap uncertainty and its limits	Reconstruct the model, work an application, test its assumptions, and explain the result in ordinary language.
8	Computational projects, testing, and communication	Reconstruct the model, work an application, test its assumptions, and explain the result in ordinary language.
9	Integration studio	Combine several chapters in one case, preserve their assumptions, and reconcile the result across representations.
10	Cumulative review	Retrieve the full course map from memory and solve mixed problems without chapter labels.

Scope anchor: <https://catalog.registrar.ucla.edu/course/2025/STATS102A>. The sequence is aligned to the official catalog description and expanded into a practical ten-week study plan.

How I would use this packet

Treat each numbered section as one chapter. Read the formal explanation, pause at the student interpretation, and see whether the easy version still says the same thing. Rework the example without looking, then cover the solution in the chapter practice box. The cumulative set at the end is deliberately mixed: knowledge becomes durable when I have to choose the tool instead of being told which chapter I am in.

Contents

1	R objects, vectorization, and reliable functions	5
1.1	Rebuild the chapter from first principles	5
1.2	Details I would refuse to skip	5
1.3	How this chapter connects	6
1.4	The full working model	6
2	Tidy data, transformation, and reproducibility	7
2.1	Rebuild the chapter from first principles	7
2.2	Details I would refuse to skip	8
2.3	How this chapter connects	8
2.4	The full working model	8
3	Statistical graphics as model checks	10
3.1	Rebuild the chapter from first principles	10
3.2	Details I would refuse to skip	10
3.3	How this chapter connects	11
3.4	The full working model	11
4	Numerical root finding and optimization	12
4.1	Rebuild the chapter from first principles	12
4.2	Details I would refuse to skip	13
4.3	How this chapter connects	13
4.4	The full working model	13
5	Random-number generation and simulation design	15
5.1	Rebuild the chapter from first principles	15
5.2	Details I would refuse to skip	15
5.3	How this chapter connects	16
5.4	The full working model	16
6	Permutation and randomization tests	17
6.1	Rebuild the chapter from first principles	17
6.2	Details I would refuse to skip	18
6.3	How this chapter connects	18

6.4	The full working model	19
7	Bootstrap uncertainty and its limits	20
7.1	Rebuild the chapter from first principles	20
7.2	Details I would refuse to skip	21
7.3	How this chapter connects	21
7.4	The full working model	21
8	Computational projects, testing, and communication	22
8.1	Rebuild the chapter from first principles	23
8.2	Details I would refuse to skip	23
8.3	How this chapter connects	23
8.4	The full working model	24

CHAPTER 1 R objects, vectorization, and reliable functions

The idea

R computing depends on atomic vectors, matrices, lists, data frames, factors, indexing, recycling, missing values, environments, and functions. Vectorized work is concise only when dimensions and types remain explicit.

Rebuild the chapter from first principles

The claim I need to own is this: R computing depends on atomic vectors, matrices, lists, data frames, factors, indexing, recycling, missing values, environments, and functions. Vectorized work is concise only when dimensions and types remain explicit. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of R objects, vectorization, and reliable functions. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach `r` objects, vectorization, and reliable functions on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from the course foundations to tidy data, transformation, and reproducibility. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

The full working model

- **Core claim.** R computing depends on atomic vectors, matrices, lists, data frames, factors, indexing, recycling, missing values, environments, and functions.
- **What moves.** Vectorized work is concise only when dimensions and types remain explicit.
- **What keeps the answer honest.** The model becomes useful only when I can apply `r` objects, vectorization, and reliable functions to an unfamiliar case and defend the assumptions.
- **Mastery standard.** I can define the objects, rebuild the rule, work a fresh case, compare alternatives, and diagnose a broken solution without relying on recognition.

How I actually think about it

I inspect class, length, names, and missingness whenever R returns an answer that looks plausible too quickly.

The easy version

R computing depends on atomic vectors, matrices, lists, data frames, factors, indexing, recycling, missing values, environments, and functions.

Worked example

Question. What is dangerous about adding vectors of lengths 6 and 4 in R?

Walkthrough. R recycles the shorter vector and warns because lengths are not multiples. The result may run but misalign values, so dimensions should be validated.

Common miss

The fastest way to get this chapter wrong is to use the visible procedure without naming its assumption, units, time period, reference group, or boundary. I put that condition beside the work and verify the result independently.

Solved chapter practice

Practice 1. What is dangerous about adding vectors of lengths 6 and 4 in R?

Solution. R recycles the shorter vector and warns because lengths are not multiples. The result may run but misalign values, so dimensions should be validated.

Practice 2. Explain `r` objects, vectorization, and reliable functions to a classmate using one definition, one concrete example, one assumption, and one failure mode.

Solution. Begin with the core claim above, use the worked case as the concrete example, state the condition highlighted by the warning, and choose a failure where that condition does not hold. A complete explanation connects all four pieces instead of listing them.

Mastery practice A. Reconstruct `r` objects, vectorization, and reliable functions from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.

Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a `r` objects, vectorization, and reliable functions problem but never states a model or checks the result. Write the shortest useful review of that solution.

Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 2 Tidy data, transformation, and reproducibility

The idea

Tidy data gives each variable a column, observation a row, and observational unit a table. Joins, reshaping, grouping, parsing, factors, dates, missingness, scripts, seeds, and project-relative paths create an auditable pipeline.

Rebuild the chapter from first principles

The claim I need to own is this: Tidy data gives each variable a column, observation a row, and observational unit a table. Joins, reshaping, grouping, parsing, factors, dates, missingness, scripts, seeds, and project-relative paths create an auditable pipeline. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of tidy data, transformation, and reproducibility. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach tidy data, transformation, and reproducibility on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from `r` objects, vectorization, and reliable functions to statistical graphics as model checks. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

The full working model

- **Core claim.** Tidy data gives each variable a column, observation a row, and observational unit a table.
- **What moves.** Joins, reshaping, grouping, parsing, factors, dates, missingness, scripts, seeds, and project-relative paths create an auditable pipeline.

- **What keeps the answer honest.** The model becomes useful only when I can apply tidy data, transformation, and reproducibility to an unfamiliar case and defend the assumptions.
- **Mastery standard.** I can define the objects, rebuild the rule, work a fresh case, compare alternatives, and diagnose a broken solution without relying on recognition.

How I actually think about it

I preserve raw data and make every derived column traceable to one named transformation.

The easy version

Tidy data gives each variable a column, observation a row, and observational unit a table.

Worked example

Question. Why validate join keys before a left join?

Walkthrough. Duplicate or missing keys can silently multiply or drop logical observations. Check uniqueness, unmatched rows, expected cardinality, and row counts.

Common miss

The fastest way to get this chapter wrong is to use the visible procedure without naming its assumption, units, time period, reference group, or boundary. I put that condition beside the work and verify the result independently.

Solved chapter practice

Practice 1. Why validate join keys before a left join?

Solution. Duplicate or missing keys can silently multiply or drop logical observations. Check uniqueness, unmatched rows, expected cardinality, and row counts.

Practice 2. Explain tidy data, transformation, and reproducibility to a classmate using one definition, one concrete example, one assumption, and one failure mode.

Solution. Begin with the core claim above, use the worked case as the concrete example, state the condition highlighted by the warning, and choose a failure where that condition does not hold. A complete explanation connects all four pieces instead of listing them.

Mastery practice A. Reconstruct tidy data, transformation, and reproducibility from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.

Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a tidy data, transformation, and reproducibility problem but never states a model or checks the result. Write the shortest useful review of that solution.

Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent

check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 3 Statistical graphics as model checks

The idea

Effective graphics match marks and scales to the variable type, show distributions and uncertainty, expose missingness, and avoid perceptual distortion. Layered plots can display data, summaries, residuals, and fitted structure together.

Rebuild the chapter from first principles

The claim I need to own is this: Effective graphics match marks and scales to the variable type, show distributions and uncertainty, expose missingness, and avoid perceptual distortion. Layered plots can display data, summaries, residuals, and fitted structure together. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of statistical graphics as model checks. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.

- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach statistical graphics as model checks on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from tidy data, transformation, and reproducibility to numerical root finding and optimization. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

The full working model

- **Core claim.** Effective graphics match marks and scales to the variable type, show distributions and uncertainty, expose missingness, and avoid perceptual distortion.
- **What moves.** Layered plots can display data, summaries, residuals, and fitted structure together.
- **What keeps the answer honest.** The model becomes useful only when I can apply statistical graphics as model checks to an unfamiliar case and defend the assumptions.
- **Mastery standard.** I can define the objects, rebuild the rule, work a fresh case, compare alternatives, and diagnose a broken solution without relying on recognition.

How I actually think about it

My shorthand for this chapter is: a plot earns space when it answers a question or reveals a model failure, not when it merely decorates a report.

The easy version

Effective graphics match marks and scales to the variable type, show distributions and uncertainty, expose missingness, and avoid perceptual distortion.

Worked example

Question. Why is a residual-versus-fitted plot useful?

Walkthrough. Curvature suggests missing functional form, a funnel suggests changing variance, and isolated points suggest unusual cases. It directly checks structure left unexplained by the fit.

Common miss

The fastest way to get this chapter wrong is to use the visible procedure without naming its assumption, units, time period, reference group, or boundary. I put that condition beside the

work and verify the result independently.

Solved chapter practice

Practice 1. Why is a residual-versus-fitted plot useful?

Solution. Curvature suggests missing functional form, a funnel suggests changing variance, and isolated points suggest unusual cases. It directly checks structure left unexplained by the fit.

Practice 2. Explain statistical graphics as model checks to a classmate using one definition, one concrete example, one assumption, and one failure mode.

Solution. Begin with the core claim above, use the worked case as the concrete example, state the condition highlighted by the warning, and choose a failure where that condition does not hold. A complete explanation connects all four pieces instead of listing them.

Mastery practice A. Reconstruct statistical graphics as model checks from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.

Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a statistical graphics as model checks problem but never states a model or checks the result. Write the shortest useful review of that solution.

Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 4 Numerical root finding and optimization

The idea

Bisection uses a sign-changing bracket for guaranteed convergence; Newton and secant methods use local slope information for speed. Optimization requires objectives, gradients or searches, stopping rules, scaling, and checks for boundaries or local solutions.

Rebuild the chapter from first principles

The claim I need to own is this: Bisection uses a sign-changing bracket for guaranteed convergence; Newton and secant methods use local slope information for speed. Optimization requires objectives, gradients or searches, stopping rules, scaling, and checks for boundaries or local solutions. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of numerical root finding and optimization. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach numerical root finding and optimization on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from statistical graphics as model checks to random-number generation and simulation design. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

The full working model

- **Core claim.** Bisection uses a sign-changing bracket for guaranteed convergence; Newton and secant methods use local slope information for speed.
- **What moves.** Optimization requires objectives, gradients or searches, stopping rules, scaling, and checks for boundaries or local solutions.

- **What keeps the answer honest.** The model becomes useful only when I can apply numerical root finding and optimization to an unfamiliar case and defend the assumptions.
- **Mastery standard.** I can define the objects, rebuild the rule, work a fresh case, compare alternatives, and diagnose a broken solution without relying on recognition.

How I actually think about it

I plot or bracket the objective before trusting an optimizer's success message.

The easy version

Bisection uses a sign-changing bracket for guaranteed convergence; Newton and secant methods use local slope information for speed.

Worked example

Question. Bisection starts with $f(a)$ and $f(b)$ of opposite signs. What invariant is preserved?

Walkthrough. At every iteration the retained interval continues to bracket at least one root under continuity, so its width halves and the root remains enclosed.

Common miss

The fastest way to get this chapter wrong is to use the visible procedure without naming its assumption, units, time period, reference group, or boundary. I put that condition beside the work and verify the result independently.

Solved chapter practice

Practice 1. Bisection starts with $f(a)$ and $f(b)$ of opposite signs. What invariant is preserved?

Solution. At every iteration the retained interval continues to bracket at least one root under continuity, so its width halves and the root remains enclosed.

Practice 2. Explain numerical root finding and optimization to a classmate using one definition, one concrete example, one assumption, and one failure mode.

Solution. Begin with the core claim above, use the worked case as the concrete example, state the condition highlighted by the warning, and choose a failure where that condition does not hold. A complete explanation connects all four pieces instead of listing them.

Mastery practice A. Reconstruct numerical root finding and optimization from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.

Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a numerical root finding and optimization problem but never states a model or checks the result. Write the shortest useful review of that solution.

Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 5 Random-number generation and simulation design

The idea

Pseudo-random uniforms feed inverse-CDF, transformation, rejection, and composition methods. Monte Carlo estimates expectations through repeated draws; seed control, diagnostics, vectorization, and Monte Carlo standard error make simulation reproducible.

Rebuild the chapter from first principles

The claim I need to own is this: Pseudo-random uniforms feed inverse-CDF, transformation, rejection, and composition methods. Monte Carlo estimates expectations through repeated draws; seed control, diagnostics, vectorization, and Monte Carlo standard error make simulation reproducible. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of random-number generation and simulation design. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical

example. If one representation disagrees with another, that disagreement is useful evidence.

- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach random-number generation and simulation design on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from numerical root finding and optimization to permutation and randomization tests. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

The full working model

- **Core claim.** Pseudo-random uniforms feed inverse-CDF, transformation, rejection, and composition methods.
- **What moves.** Monte Carlo estimates expectations through repeated draws; seed control, diagnostics, vectorization, and Monte Carlo standard error make simulation reproducible.
- **What keeps the answer honest.** The model becomes useful only when I can apply random-number generation and simulation design to an unfamiliar case and defend the assumptions.
- **Mastery standard.** I can define the objects, rebuild the rule, work a fresh case, compare alternatives, and diagnose a broken solution without relying on recognition.

How I actually think about it

My shorthand for this chapter is: a simulation answer needs its own uncertainty estimate; more digits from the computer are not more evidence.

The easy version

Pseudo-random uniforms feed inverse-CDF, transformation, rejection, and composition methods.

Worked example

Question. A simulated probability is 0.40 from 10,000 Bernoulli trials. Approximate Monte Carlo standard error.

Walkthrough. It is $\sqrt{0.4 \cdot 0.6 / 10000}$, about 0.0049. Repeating with four times as many draws would roughly halve this error.

Common miss

The fastest way to get this chapter wrong is to use the visible procedure without naming its assumption, units, time period, reference group, or boundary. I put that condition beside the work and verify the result independently.

Solved chapter practice

Practice 1. A simulated probability is 0.40 from 10,000 Bernoulli trials. Approximate Monte Carlo standard error.

Solution. It is $\sqrt{0.4 \cdot 0.6 / 10000}$, about 0.0049. Repeating with four times as many draws would roughly halve this error.

Practice 2. Explain random-number generation and simulation design to a classmate using one definition, one concrete example, one assumption, and one failure mode.

Solution. Begin with the core claim above, use the worked case as the concrete example, state the condition highlighted by the warning, and choose a failure where that condition does not hold. A complete explanation connects all four pieces instead of listing them.

Mastery practice A. Reconstruct random-number generation and simulation design from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.

Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a random-number generation and simulation design problem but never states a model or checks the result. Write the shortest useful review of that solution.

Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 6 Permutation and randomization tests**The idea**

Randomization tests generate a null distribution by applying transformations justified by the design, such as permuting treatment labels under exchangeability. The statistic, alternative, and assignment mechanism must match the scientific hypothesis.

Rebuild the chapter from first principles

The claim I need to own is this: Randomization tests generate a null distribution by applying transformations justified by the design, such as permuting treatment labels under exchangeability. The statistic,

alternative, and assignment mechanism must match the scientific hypothesis. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of permutation and randomization tests. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach permutation and randomization tests on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from random-number generation and simulation design to bootstrap uncertainty and its limits. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

The full working model

- **Core claim.** Randomization tests generate a null distribution by applying transformations justified by the design, such as permuting treatment labels under exchangeability.
- **What moves.** The statistic, alternative, and assignment mechanism must match the scientific hypothesis.
- **What keeps the answer honest.** The model becomes useful only when I can apply permutation and randomization tests to an unfamiliar case and defend the assumptions.
- **Mastery standard.** I can define the objects, rebuild the rule, work a fresh case, compare alternatives, and diagnose a broken solution without relying on recognition.

How I actually think about it

I shuffle only what the null and design make exchangeable; arbitrary shuffling is not a valid test.

The easy version

Randomization tests generate a null distribution by applying transformations justified by the design, such as permuting treatment labels under exchangeability.

Worked example

Question. In a paired experiment, what should be randomized under a sharp no-effect null?

Walkthrough. Randomly flip the treatment-control sign within each pair rather than permuting individual observations across pairs, preserving the paired assignment structure.

Common miss

The fastest way to get this chapter wrong is to use the visible procedure without naming its assumption, units, time period, reference group, or boundary. I put that condition beside the work and verify the result independently.

Solved chapter practice

Practice 1. In a paired experiment, what should be randomized under a sharp no-effect null?

Solution. Randomly flip the treatment-control sign within each pair rather than permuting individual observations across pairs, preserving the paired assignment structure.

Practice 2. Explain permutation and randomization tests to a classmate using one definition, one concrete example, one assumption, and one failure mode.

Solution. Begin with the core claim above, use the worked case as the concrete example, state the condition highlighted by the warning, and choose a failure where that condition does not hold. A complete explanation connects all four pieces instead of listing them.

Mastery practice A. Reconstruct permutation and randomization tests from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.

Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects

and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a permutation and randomization tests problem but never states a model or checks the result. Write the shortest useful review of that solution.

Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 7 Bootstrap uncertainty and its limits

The idea

The bootstrap resamples observed units to approximate an estimator's sampling distribution. Percentile, basic, and studentized intervals differ; dependence, small samples, extreme statistics, and biased sampling require modified methods or caution.

Rebuild the chapter from first principles

The claim I need to own is this: The bootstrap resamples observed units to approximate an estimator's sampling distribution. Percentile, basic, and studentized intervals differ; dependence, small samples, extreme statistics, and biased sampling require modified methods or caution. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of bootstrap uncertainty and its limits. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach bootstrap uncertainty and its limits on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from permutation and randomization tests to computational projects, testing, and communication. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

The full working model

- **Core claim.** The bootstrap resamples observed units to approximate an estimator's sampling distribution.
- **What moves.** Percentile, basic, and studentized intervals differ; dependence, small samples, extreme statistics, and biased sampling require modified methods or caution.
- **What keeps the answer honest.** The model becomes useful only when I can apply bootstrap uncertainty and its limits to an unfamiliar case and defend the assumptions.
- **Mastery standard.** I can define the objects, rebuild the rule, work a fresh case, compare alternatives, and diagnose a broken solution without relying on recognition.

How I actually think about it

I resample the unit that was independently sampled, not whichever row is easiest to code.

The easy version

The bootstrap resamples observed units to approximate an estimator's sampling distribution.

Worked example

Question. How should clustered classroom data be bootstrapped?

Walkthrough. Resample classrooms as whole clusters, carrying their students together. Row-wise resampling would break within-class dependence and understate uncertainty.

Common miss

The fastest way to get this chapter wrong is to use the visible procedure without naming its assumption, units, time period, reference group, or boundary. I put that condition beside the work and verify the result independently.

Solved chapter practice

Practice 1. How should clustered classroom data be bootstrapped?

Solution. Resample classrooms as whole clusters, carrying their students together. Row-wise resampling would break within-class dependence and understate uncertainty.

Practice 2. Explain bootstrap uncertainty and its limits to a classmate using one definition, one concrete example, one assumption, and one failure mode.

Solution. Begin with the core claim above, use the worked case as the concrete example, state the condition highlighted by the warning, and choose a failure where that condition does not hold. A complete explanation connects all four pieces instead of listing them.

Mastery practice A. Reconstruct bootstrap uncertainty and its limits from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.

Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a bootstrap uncertainty and its limits problem but never states a model or checks the result. Write the shortest useful review of that solution.

Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 8 Computational projects, testing, and communication

The idea

A statistical computing project separates data ingestion, validation, analysis, figures, and reporting. Assertions, unit tests, profiling, version control, session information, readable functions, and literate documents make results reproducible and maintainable.

Rebuild the chapter from first principles

The claim I need to own is this: A statistical computing project separates data ingestion, validation, analysis, figures, and reporting. Assertions, unit tests, profiling, version control, session information, readable functions, and literate documents make results reproducible and maintainable. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of computational projects, testing, and communication. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach computational projects, testing, and communication on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from bootstrap uncertainty and its limits to the cumulative course model. The earlier material supplies the language and constraints; this chapter adds a new reasoning

move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

The full working model

- **Core claim.** A statistical computing project separates data ingestion, validation, analysis, figures, and reporting.
- **What moves.** Assertions, unit tests, profiling, version control, session information, readable functions, and literate documents make results reproducible and maintainable.
- **What keeps the answer honest.** The model becomes useful only when I can apply computational projects, testing, and communication to an unfamiliar case and defend the assumptions.
- **Mastery standard.** I can define the objects, rebuild the rule, work a fresh case, compare alternatives, and diagnose a broken solution without relying on recognition.

How I actually think about it

My goal is a project another student can run from a clean session without guessing what I clicked.

The easy version

A statistical computing project separates data ingestion, validation, analysis, figures, and reporting.

Worked example

Question. What should a reproducibility check do?

Walkthrough. Start from documented inputs in a clean environment, run the scripted pipeline, confirm tests and key outputs, record package versions, and verify that no hidden manual state is required.

Common miss

The fastest way to get this chapter wrong is to use the visible procedure without naming its assumption, units, time period, reference group, or boundary. I put that condition beside the work and verify the result independently.

Solved chapter practice

Practice 1. What should a reproducibility check do?

Solution. Start from documented inputs in a clean environment, run the scripted pipeline, confirm tests and key outputs, record package versions, and verify that no hidden manual state is required.

Practice 2. Explain computational projects, testing, and communication to a classmate using one definition, one concrete example, one assumption, and one failure mode.

Solution. Begin with the core claim above, use the worked case as the concrete example, state the condition highlighted by the warning, and choose a failure where that condition does not hold. A complete explanation connects all four pieces instead of listing them.

Mastery practice A. Reconstruct computational projects, testing, and communication from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.

Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a computational projects, testing, and communication problem but never states a model or checks the result. Write the shortest useful review of that solution.

Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

Cumulative study plan

I use this packet in three passes. First I reconstruct each model and work its examples without looking. Second I mix chapters so the tool is no longer named for me. Third I explain a complete case aloud, including assumptions, units, uncertainty, failure modes, and what evidence would change my conclusion.