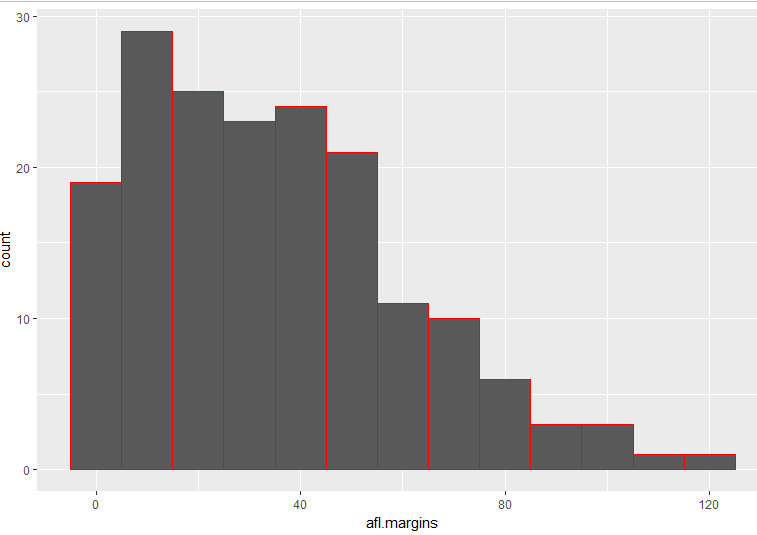
***Learning Statistics with R - University of Adelaide***

***Part IV - Working with Data***

**Chapter 5 - Descriptive Stats**

**Chapter 5.1 – Measures of Center**

* Any time you get a new data set to look at, 1 of the first tasks to do is find ways of summarizing the data in a compact, easily-understood fashion.
* This is what **descriptive statistics** (as opposed to **inferential statistics**) is all about.
* In fact, to many people, the term "statistics" is synonymous w/ descriptive statistics.



* This histogram is **right/positively-skewed** (*tail to the right*)
* This indicates that higher margins are less frequent = use median as center of measure/measure of central tendency b/c w/ a skewed distribution, the mean is pulled to the side of the tail (to the right)
* **Mean** = center of mass/gravity of a data set (balancing point if histogram was on a seesaw)
* If data are **nominal**, *don’t* use mean OR median, since both rely on the idea that *the numbers assigned to values are meaningful*
* If the numbering scheme is *arbitrary*, it is probably best to use **mode** instead.
* If data are **ordinal,** we’re more likely to want to use **median**, which only makes use of the ordering info within data (i.e., which numbers are bigger), but does not depend on the *precise* numbers involved
* **Mean**, on the other hand, makes use of the *precise numeric values* assigned to observations, so it is not really appropriate for ordinal data.
* For **interval** and **ratio** data, either mean or median is generally acceptable, depending on what you are trying to achieve.
* mean advantage = it uses ALL the info in the data (useful when you don’t have a lot of data)
* mean disadvantage = *very sensitive* to extreme values (**outliers**)
* There’re systematic differences between mean + median when a distribution is *asymmetric/skewed*.
* Median is closer to the "body" of the distribution/histogram, whereas mean is dragged towards the tail (extreme values)
* 1 of the fundamental rules of applied statistics = **the data are messy**, since real life is never simple, so data sets you obtain are never as straightforward as the statistical theory says.
* **Robust statistics** = tries to grapple w/ the messiness of real data + develop theory to cope w/ it.
* Ex: 100, 2, 3, 4, 5, 6, 7, 8, 9, 10 --> If observed in real life data set, probably suspect something funny is going on w/ the “100”, it’s probably an outlier
* Consider removing it from the data set
* In real life, however, you don't always get such cut-and-dried examples.
* For instance, we might get (15, 2, 3, 4, 5, 6, 7, 8, 9, 12), and 15 may look suspicious, but nowhere as much as 100
* It’s little trickier, as it might be a legitimate observation, it might not.
* When faced with a situation where some of the most extreme-valued observations might not be quite trustworthy, the mean is not necessarily a good measure of central tendency
* b/c it’s highly sensitive to even just 1 or 2 extreme values, + is thus NOT considered to be a **robust measure**
* 1 remedy to this 🡺 *Use the median*
* For more general solution = use a **trimmed mean** -🡪 "discard" the most extreme examples on both ends (largest + smallest values) of the distribution and take the mean what remains.
* Goal of trimmed mean = preserve best characteristics of the mean + median
* It’s not highly influenced by extreme outliers, but like the mean, it uses more than 1 observation
* We generally describe a trimmed mean in terms of the % of observations on either side that’re discarded.
* Ex: **10% trimmed mean** discards largest + smallest 10% of observations + takes the mean of the remaining 80%
* 0% trimmed mean = regular mean, 50% trimmed mean = median.
* In this sense, a trimmed means provide a whole family of central tendency measures that span the range from regular mean to the median
* If there’s a fairly substantial difference between mean and median, the mean may be influenced a bit too much by extreme values
* **Mode** of dataset = value of the variable that makes up the mode
* **Modal frequency =** actual number of occurrences of the mode.'
* Mode is most often calculated for **nominal** data (b/c means + medians are useless for those variables)
* Plus, there are some situations in which you really DO want to know the mode of an ordinal, such as **interval** or **ratio** scale variable.
* Ex: In the afl.margins vector, the values clearly indicate a **ratio** (i.e. point differential field), so in most situations mean or median is the measure of central tendency we want.
* Consider a friend of yours is offering a bet to pick a football game at random + you have to guess the exact margin.
* If you guess correctly, win $50, if not, lose $1
* For this bet, mean + median are completely useless, + it is the mode you should bet on

**Chapter 5.2 - Measures of Variability**

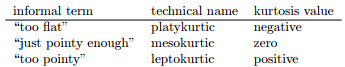
* Statistics discussed so far all relate to **central tendency**, which all talk about which values are "in the middle" or "popular" in the data.
* Central tendency is not the only type of summary statistic to calculate,
* The 2nd thing we want is a **measure of the variability** of the data, or how *spread out* are the data, how *f*ar away from the mean/median do observed values tend to be?
* Although range is the *simplest* way to quantify variability, it's one of the *worst* b/c we want our summary measure to be robust
* If the data has 1 or 2 extremely bad values in it, we would like our stats not to be unduly influenced by these cases, but unfortunately range is
* Ex: (100, 2, 3, 4, 5, 6, 7, 8, 9, 10) 🡺 range = 110 and *is not robust*, but if the outlier 100 were removed, we would have a range of only 8.
* **Interquartile Range (IQR)** is like the range, but instead the difference between biggest + smallest value, its difference between the 25th quartile/percentiles (Q1) + 75th quartile/percentiles (Q3).
* Reminder: 10th percentile of a data set = smallest number, x, such that 10% of data is less than x
* So, median = 50th quartile/percentile (2nd quartile, Q2)
* It’s bbvious how to interpret the range, a little less obvious to interpret IQR.
* Simplest way to think about it: **IQR** = range spanned by the *middle half* of the data
* 1/4 of the data falls both below + above the 25th + 75th percentiles, leaving the "middle half" of the data between the 2.
* The 2 measures so far, range + IQR, both rely on the idea that we can measure the spread of the data by looking at the quartiles of the data.
* However, this isn't the only way to think about the problem.
* A different approach is to select a meaningful *reference point* (usually mean or median) + then report the "typical" deviations from that reference point.
* A typical deviation is usually the mean or median value *of these deviations*
* In practice, this leads to 2 different measures = **mean absolute deviation** + **median absolute deviation**
* The measure based on the median, **median absolute deviation**, seems to be used in statistics, + does seem to be the better of the 2
* The measure based on the mean, **mean absolute deviation**, does occasionally show up in psychology though.
* Although the mean absolute deviation measure has its uses, it's not the best measure of variability to use.
* From a purely *mathematical* perspective, there are some solid reasons to prefer *squared* deviations rather than absolute deviations
* This obtains a measure, **variance, s^2**, which has a lot of really nice statistical properties but 1 massive psychological flaw
* It’s basically same formula as mean absolute deviation, except w/ squares of deviations rather than absolute values
* **NOTE**: Variances are **additive**
* X and Y have variances VarX and VarY
* Define a new variable Z that = sum of X + Y
* It turns out **VarZ = VarX + VarY** is a very useful property (but not true of other measures)
* There is a subtle distinction between describing a sample + making guesses about a population from which the sample came
* Up to this point, it's been a distinction without a difference.
* Regardless of whether describing a sample or drawing inferences about a population, *mean is calculated the same*
* NOT SO for variance or standard deviation, or for many other measures
* Most of the time we’re not interested in the sample in + of itself but we have a sample exist to tell you something about the world.
* In other words, we’re starting to move away from a \*\***statistic**\*\* to a \*\***parameter**\*\*.
* How do you interpret the variance?
* Descriptive statistics are supposed to describe things + right now the variance is a gibberish number.
* *There really is no human-friendly interpretation of variance, which is the most serious problem w/ it*
* Although it has some elegant mathematical properties that suggest it is a fundamental quantity for expressing variation, it is completely useless if you want to communicate w/ an actual human.
* Variances are completely uninterpretable *in terms of the original variable*
* If all the numbers have been squared, they do not mean anything anymore, which is a huge issue.
* Ex: Game 1 Margin was 376.36 points-squared higher than the average margin
* There is not a real unit of measurement
* Suppose you like using variance because of mathematical properties not talked about yet
* But we want a measure expressed in the *same units as the data itself* (i.e. points, not points-squared)
* Solution: Take the *square root of the variance* = \*\***standard deviation**\*\* = \*\***root mean squared deviation (RMSD)**\*\*
* "A standard deviation of 18.01 points" is much easier to understand since it's expressed in the original units
* Interpreting SDs is slightly more complex b/c SD is derived from variance, a quantity w/ little to no meaning to humans
* As a consequence, we rely on a simple rule of thumb:
* In general, expect 68% of data to fall w/in 1 SD of the mean, 95% to fall within 2 SD, and 99.7% to fall within 3 SD
* This tends to work pretty well most of the time, but it's not exact + is calculated *based on an assumption that the data is symmetric/normal/bell-shaped* (AFL winning margins is not)
* In this case, 65.3% of the AFL margins lies within 1 SD, which is still pretty consistent w/ the "approximately 68%” rule
* Back to the Median absolute deviation (MAD), which is pretty much identical to the idea behind the mean absolute deviation (AAD) but you use the median instead
* It actually has a straightforward interpretation*: Every observation in the data lies some distance away from the typical value* (here, the median).
* The MAD is an attempt to *describe a typical deviation from a typical value in the data set.*
* It wouldn't be unreasonable to interpret the MAD value of 19.5 for AFL by saying:
* "The median winning margin in 2010 was 30.5, indicating a typical game involved a winning margin of about 30 points. However, there was a fair amount of variation from game to game, w/ a MAD value = 19.5, indicating a typical winning margin would differ from this median value by about 19-20 points.'
* Although a "raw" MAD value is completely interpretable on its own terms, it is not actually how it's used in a lot of real world contexts.
* Instead, a researcher *actually* wants to calculate the *SD*, but since the mean is very sensitive to extreme values, so too is the SD
* So, just like using the median as a "robust" way of calculating "something that is like the mean", it's not uncommon to use MAD as a method for calculating "something that is like the SD".
* Unfortunately, a raw MAD value of 19.5 doesn't do this when if our SD = 26.07.
* However under certain assumptions, we can *multiply the raw MAD value by 1.4826* + obtain a number that is directly comparable to the SD
* As a consequence, a default value of this constant = 1.4826
* **NOTE:** If you want to use this "corrected" MAD value as a robust version of the SD, you really are relying on the assumption that the data is normal (which is not true for afl.margins)
* Summary\*\*
* **Range** = full spread of data, very vulnerable to outliers, + as a consequence is not often used unless you have good reasons to care about extremes in the data.
* **IQR** = where the "middle half" of the data sits, is pretty robust, complements median nicely
* **Mean absolute deviation (AAD)** = how far observations are on average from the mean, is very interpretable, but has a few minor issues (not discussed here) that make it less attractive to statisticians than the SD, so it is used sometimes, but not often.
* **Variance** = average squared deviation from the mean, mathematically elegant, is probably the "right" way to describe variation around a mean, but is completely uninterpretable b/c it does not use the same units as the data, so almost never used except as a mathematical tool but is buried "under the hood" of a very large number of statistical tools.
* **SD** = square root of the variance, fairly elegant mathematically, expressed in same units as data, so is interpreted pretty well, in situations where mean is the measure of central tendency, is the default + by far the most popular measure of variation.
* **Median absolute deviation (MAD)** = typical (i.e. median) deviation from the median value, is simple + interpretable in *raw* form, is a robust way to estimate SD in the *corrected* form, used for *\*some\** kinds of data sets, not used very often, but does get reported sometimes.
* In short, the **IQR** and the **SD** are easily the 2 most common measures used to report variability of data, but there’re situations in which the others are used.

**Chapter 5.3 – Skew and Kurtosis**

* In practice, neither skew nor kurtosis is used anywhere near as frequently as the measures of central tendency + variability
* \*\***Skew**\*\* is pretty important, so you see it mentioned a fair bit; but \*\***kurtosis**\*\* is rarely reported in a scientific manner
* \*\***Skewness**\*\* = a measure of asymmetry
* Data w/ a lot of extreme small values 🡪 lower tail is "longer" than upper tail w/ few extremely large values = **negatively/left skewed.**
* More extremely large values than extremely small ones = **positively/right skewed**.
* Actual formula for the skewness of a data set:



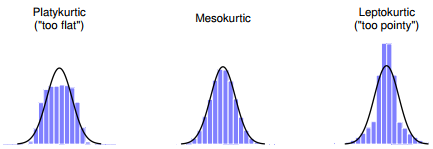
* Not surprisingly, AFL winning margins data is fairly skewed, as we saw before in the histogram
* \*\***Kurtosis**\*\* = a measure of the "**pointiness**" of a data set
* By convention, the **normal curve** has 0 kurtosis, or is \*\***mesokurtic** (just pointy enough)
* So pointiness of a data set is assessed *relative to the normal curve.*
* Not pointy enough (bit of a uniform distribution) = negative kurtosis = \*\***platykurtic**\*\*
* Too pointy (large peak relative to all else) = kurtosis = positive = \*\***leptokurtic**\*\*



* The equation for kurtosis is pretty similar in spirit to formulas seen already for variance + skewness:

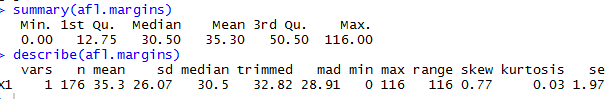


* Afl.margins is close to **mesokurtic** = just pointy enough (0.2962)

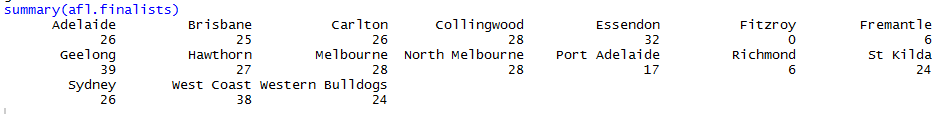


**Chapter 5.4 – Summarizing a Dataset**

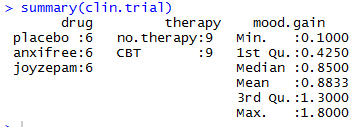
* It’s kind of annoying to have to separately calculate means, medians, standard deviations, skews etc.
* R has some helpful functions that would do all these tedious calculations at once






* Helpful to declare a nominal scale variable as a *factor* rather than a *character* vector
* Okay what about data frames?
* When you pass a data frame to summary() function, it produces a slightly condensed summary of each variable inside the data frame



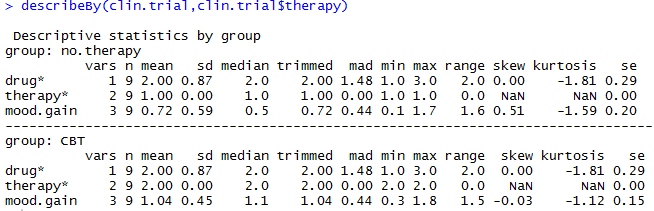
* Describe() output:
* **n =** sample size (more precisely, the number of non-missing values)
* **mean =** sample mean
* **sd** = the (bias corrected) standard deviation
* **median**
* **trimmed** = 10% trimmed mean (default)
* **mad =** median absolute deviation
* **min**
* **max**
* **range**
* **skew**
* **kurtosis**
* **se** = standard error of the mean
* describe() function (in the psych package) is a little different, and it’s really only intended to be useful when data are interval or ratio scale (encoded as numeric vectors).
* For nominal or ordinal variables (encoded as factors vectors), most of these descriptive statistics are not all that useful
* describe() converts factors + logical variables to numeric vectors in order to do the calculations.
* Such variables are marked w/ \*’s and most of the time, the descriptive statistics for those variables won’t make much sense.
* If you try to feed it a data frame that includes a character vector as a variable, it produces an error.



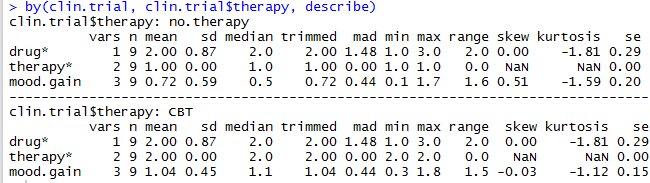
* Output for the asterisked variables is pretty meaningless, and should be ignored.
* However, for the mood.gain variable, there’s a lot of useful information.

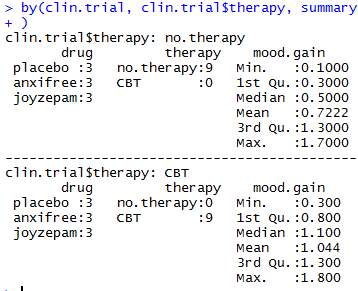
**Chapter 5.5 – Descriptive Statistics Separately For Each Group**

* It is very commonly the case you find yourself needing to look at descriptive statistics, broken down by some grouping variable

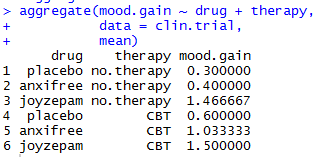
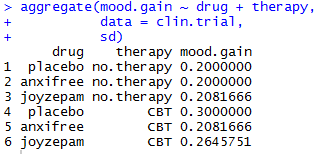


* See 2 outputs, 1 for the CBT group + 1 for the “no therapy” group.
* output still displays asterisks for factor variables
* Nevertheless, this has given us some really useful descriptive statistics for mood.gain, broken down as a function of therapy.
* Somewhat more general solution is offered by the by() function





* What if you have multiple grouping variables?
* Suppose, for example, you would like to look at average mood gain separately for all possible combinations of drug + therapy.
* Possible w/ by() and describeBy() functions used in conjunction, but it’s usually more convenient to use aggregate()

**Chapter 5.6 - Standard Scores**

* Suppose my friend is putting together a new questionnaire intended to measure “grumpiness”.” w/ 50 questions, which you can answer in a grumpy way or not.
* Across a big sample (hypothetically, 1M people or so!), the data are fairly normally distributed, w/ mean grumpiness score = 17/50 questions answered in a grumpy way + the SD = 5.
* In contrast, when *I* take the questionnaire, I answer 35/50 questions in a grumpy way.
* So, how grumpy am I?
* 1 way to think about would be to say I have *grumpiness* of 35/50, so you might say I’m 70% grumpy
* But that’s a bit weird, when you think about it.
* If my friend had phrased questions a bit differently, people might have answered in a different way, so the overall distribution of answers could easily move up/down depending on the precise way in which questions were asked.
* So, I’m only 70% grumpy with respect to *this set of survey questions.*
* Even if it’s a very good questionnaire, this isn’t very a informative statement.
* A simpler way around this is to describe my grumpiness *by comparing me to other people*.
* Out of the sample of 1M people, only 159 people were as grumpy as me, suggesting I’m in the top 0.016% of people for grumpiness.
* This makes much more sense than trying to interpret raw data.
* This idea that we should describe my grumpiness in terms of the *overall distribution of the grumpiness of humans* is the qualitative idea that **standardization** attempts to get at.
* 1 way to do this is to *describe everything in terms of* ***percentiles***.
* However, the problem w/ doing this is that “it’s lonely at the top”.
* Suppose that my friend had only collected a sample of 1k people (still a pretty big sample for the purposes of testing a new questionnaire) + this time got a mean 16/50 w/ SD = 5
* The problem is that almost certainly, not a single person in that sample would be as grumpy as me.
* A different approach is to convert my grumpiness score into a **standard score**, referred to as **z-score**
* The **standard z-score** = *the number of standard deviations above/below the mean that a score lies.*
* To phrase it in “pseudo-maths” the standard score is calculated like this:



* In actual math, the equation for the z-score is



* We can now transform a raw grumpiness into a standardized grumpiness score



* To interpret this value, recall the rough heuristic 99.7% of values are expected to lie w/in 3 SDs of the mean, so the fact that my grumpiness corresponds to a z score = 3.6 indicates I’m very grumpy indeed
* A function called **pnorm()** allows us to be a bit more precise than this.
* Specifically, it allows us to calculate a theoretical percentile rank for a score based on a z-score



* This output is fairly straightforward: it suggests that I’m grumpier than 99.98% of people.
* In addition to allowing you to interpret a raw score in relation to a larger population (+ thereby allowing you to make sense of variables that lie on arbitrary scales), standard scores serve a 2nd useful function.
* Standard scores can be *compared to one another in situations where the raw scores can’t.*
* Suppose my friend also had another questionnaire that measured extraversion using a 24-item questionnaire.
* The overall mean for this measure = 13 w/ SD 4 + I scored a 2
* It doesn’t make a lot of sense to try to compare a raw score of 2 on the extraversion questionnaire to a raw score of 35 on the grumpiness questionnaire.
* The raw scores for the 2 variables are “about” fundamentally different things, so this would be like comparing apples to oranges.
* If we calculate the standard scores, we get z for grumpiness = 3.6 + z for extraversion = -2.75.
* These 2 numbers CAN be compared to each other
* I’m *much less* extraverted than most people (z = -2.75) + much grumpier than most people (z = 3.6)
* But the *extent* of my unusualness is much more extreme for grumpiness (3.6 is a bigger number than 2.75).
* B/c each standardized score is a statement about where an observation falls *relative to its own population*, it is possible to compare standardized scores across completely different variables.

**Chapter 5.7 – Correlations**

* Up to this point we have focused entirely on how to construct descriptive statistics for a single variable.
* What we haven’t done is talked about how to describe the relationships between variables in the data.
* To do that, we want to talk mostly about the correlation between variables. But first, we need some
* data.
* 13I haven’t discussed how to compute z-scores, explicitly, but you can probably guess. For a variable X, the simplest way
* is to use a command like (X - mean(X)) / sd(X). There’s also a fancier function called scale() that you can use, but it relies
* on somewhat more complicated R concepts that I haven’t explained yet.
* 14Technically, b/c I’m calculating means + SDs from a sample of data, but want to talk about my
* grumpiness relative to a population, what I’m actually doing is estimating a z score. However, since we haven’t talked
* about estimation yet (see Chapter 10) I think it’s best to ignore this subtlety, especially as it makes very little difference to
* our calculations.
* 15Though some caution is usually warranted. It’s not always the case that one SD on variable A corresponds
* to the same “kind” of thing as one SD on variable B. Use common sense when trying to determine
* whether or not the z scores of two variables can be meaningfully compared.
* - 137 -
* Table 5.1: Descriptive statistics for the parenthood data.
* variable min max mean median std. dev IQR
* Dan’s grumpiness 41 91 63.71 62 10.05 14
* Dan’s hours slept 4.84 9.00 6.97 7.03 1.02 1.45
* Dan’s son’s hours slept 3.25 12.07 8.05 7.95 2.07 3.21
* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
* 5.7.1 The data
* After spending so much time looking at the AFL data, I’m starting to get bored w/ sports. Instead,
* let’s turn to a topic close to every parent’s heart: sleep. The following data set is fictitious, but based
* on real events. Suppose I’m curious to find out how much my infant son’s sleeping habits affect my
* mood. Let’s say that I can rate my grumpiness very precisely, on a scale from 0 (not at all grumpy)
* to 100 (grumpy as a very, very grumpy old man). +, lets also assume that I’ve been measuring my
* grumpiness, my sleeping patterns + my son’s sleeping patterns for quite some time now. Let’s say, for
* 100 days. +, being a nerd, I’ve saved the data as a file called parenthood.Rdata. If we load the data...
* > load( "parenthood.Rdata" )
* > who(TRUE)
* -- Name -- -- Class -- -- Size --
* parenthood data.frame 100 x 4
* $dan.sleep numeric 100
* $baby.sleep numeric 100
* $dan.grump numeric 100
* $day integer 100
* ... we see that the file contains a single data frame called parenthood, which contains four variables
* dan.sleep, baby.sleep, dan.grump + day. If we print out the data, here’s what we get:
* > parenthood
* dan.sleep baby.sleep dan.grump day
* 1 7.59 10.18 56 1
* 2 7.91 11.66 60 2
* 3 5.14 7.92 82 3
* 4 7.71 9.61 55 4
* BLAH BLAH BLAH
* 98 5.38 6.97 82 98
* 99 7.02 6.56 55 99
* 100 6.45 7.93 74 100
* Next, I’ll calculate some basic descriptive statistics:
* > describe( parenthood )
* var n mean sd median trimmed mad min max
* dan.sleep 1 100 6.97 1.02 7.03 7.00 1.09 4.84 9.00
* baby.sleep 2 100 8.05 2.07 7.95 8.05 2.33 3.25 12.07
* dan.grump 3 100 63.71 10.05 62.00 63.16 9.64 41.00 91.00
* day 4 100 50.50 29.01 50.50 50.50 37.06 1.00 100.00
* - 138 -
* My grumpiness
* Frequency
* 40 50 60 70 80 90
* 0 5 10 15 20
* My sleep (hours)
* Frequency
* 5 6 7 8 9
* 0 5 10 15 20
* The baby’s sleep (hours)
* Frequency
* 4 6 8 10 12
* 0 5 10 15 20
* (a) (b) (c)
* Figure 5.6: Histograms for the three interesting variables in the parenthood data set.
* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
* range skew kurtosis se
* dan.sleep 4.16 -0.29 -0.65 0.10
* baby.sleep 8.82 -0.02 -0.61 0.21
* dan.grump 50.00 0.43 -0.04 1.00
* day 99.00 0.00 -1.20 2.90
* Finally, to give a graphical depiction of what each of the three interesting variables looks like, Figure 5.6
* plots histograms. One thing to note: just b/c R can calculate dozens of different statistics doesn’t
* mean you should report all of them. If I were writing this up for a report, I’d probably pick out those
* statistics that are of most interest to me (+ to my readership), + then put them into a nice, simple
* table like the one in Table 5.1.
* 16 Notice that when I put it into a table, I gave everything “human
* readable” names. This is always good practice. Notice also that I’m not getting enough sleep. This isn’t
* good practice, but other parents tell me that it’s standard practice.
* 5.7.2 The strength + direction of a relationship
* We can draw scatterplots to give us a general sense of how closely related two variables are. Ideally though,
* we might want to say a bit more about it than that. For instance, let’s compare the relationship between
* dan.sleep + dan.grump (Figure 5.7, left) w/ that between baby.sleep + dan.grump (Figure 5.7, right).
* When looking at these two plots side by side, it’s clear that the relationship is qualitatively the same in
* both cases: more sleep equals less grump! However, it’s also pretty obvious that the relationship between
* dan.sleep + dan.grump is stronger than the relationship between baby.sleep + dan.grump. The plot
* on the left is “neater” than the one on the right. What it feels like is that if you want to predict what
* my mood is, it’d help you a little bit to know how many hours my son slept, but it’d be more helpful to
* know how many hours I slept.
* In contrast, let’s consider the two scatterplots shown in Figure 5.8. If we compare the scatterplot
* of “baby.sleep v dan.grump” (left) to the scatterplot of “‘baby.sleep v dan.sleep” (right), the overall
* strength of the relationship is the same, but the direction is different. That is, if my son sleeps more,
* 16Actually, even that table is more than I’d bother w/. In practice most people pick one measure of central tendency,
* + one measure of variability only.
* - 139 -
* 5 6 7 8 9
* 40 50 60 70 80 90
* My sleep (hours)
* My grumpiness
* 4 6 8 10 12
* 40 50 60 70 80 90
* The baby’s sleep (hours)
* My grumpiness
* (a) (b)
* Figure 5.7: Scatterplots showing the relationship between dan.sleep + dan.grump (left) + the relationship
* between baby.sleep + dan.grump (right).
* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
* I get more sleep (positive relationship, right hand side), but if he sleeps more then I get less grumpy
* (negative relationship, left hand side).
* 5.7.3 The correlation coefficient
* We can make these ideas a bit more explicit by introducing the idea of a correlation coefficient (or,
* more specifically, Pearson’s correlation coefficient), which is traditionally denoted by r. The correlation
* coefficient between two variables X + Y (sometimes denoted rXY ), which we’ll define more precisely
* in the next section, is a measure that varies from ´1 to 1. When r “ ´1 it means that we have a perfect
* negative relationship, + when r “ 1 it means we have a perfect positive relationship. When r “ 0,
* there’s no relationship at all. If you look at Figure 5.9, you can see several plots showing what different
* correlations look like.
* The formula for the Pearson’s correlation coefficient can be written in several different ways. I think
* the simplest way to write down the formula is to break it into two steps. Firstly, let’s introduce the idea
* of a covariance. The covariance between two variables X + Y is a generalisation of the notion of the
* variance; it’s a mathematically simple way of describing the relationship between two variables that isn’t
* terribly informative to humans:
* CovpX, Y q “ 1
* N ´ 1
* ÿ
* N
* i“1
* `
* Xi ´ X¯
* ˘ `Yi ´ Y¯
* ˘
* B/c we’re multiplying (i.e., taking the “product” of) a quantity that depends on X by a quantity
* that depends on Y + then averaging17, you can think of the formula for the covariance as an “average
* cross product” between X + Y . The covariance has the nice property that, if X + Y are entirely
* unrelated, then the covariance is exactly zero. If the relationship between them is positive (in the
* 17Just like we saw w/ the variance + the SD, in practice we divide by N ´ 1 rather than N.
* - 140 -
* 4 6 8 10 12
* 40 50 60 70 80 90
* The baby’s sleep (hours)
* My grumpiness
* 4 6 8 10 12
* 5 6 7 8 9
* The baby’s sleep (hours)
* My sleep (hours)
* (a) (b)
* Figure 5.8: Scatterplots showing the relationship between baby.sleep + dan.grump (left), as compared
* to the relationship between baby.sleep + dan.sleep (right).
* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
* sense shown in Figure 5.9) then the covariance is also positive; + if the relationship is negative then
* the covariance is also negative. In other words, the covariance captures the basic qualitative idea of
* correlation. Unfortunately, the raw magnitude of the covariance isn’t easy to interpret: it depends on
* the units in which X + Y are expressed, + worse yet, the actual units that the covariance itself
* is expressed in are really weird. For instance, if X refers to the dan.sleep variable (units: hours) +
* Y refers to the dan.grump variable (units: grumps), then the units for their covariance are “hours ˆ
* grumps”. + I have no freaking idea what that would even mean.
* The Pearson correlation coefficient r fixes this interpretation problem by standardising the covariance,
* in pretty much the exact same way that the z-score standardises a raw score: by dividing by the standard
* deviation. However, b/c we have two variables that contribute to the covariance, the standardisation
* only works if we divide by both SDs.18 In other words, the correlation between X + Y
* can be written as follows:
* rXY “
* CovpX, Y q
* σˆX σˆY
* By doing this standardisation, not only do we keep all of the nice properties of the covariance discussed
* earlier, but the actual values of r are on a meaningful scale: r “ 1 implies a perfect positive relationship,
* + r “ ´1 implies a perfect negative relationship. I’ll expand a little more on this point later, in
* Section 5.7.5. But before I do, let’s look at how to calculate correlations in R.
* 5.7.4 Calculating correlations in R
* Calculating correlations in R can be done using the cor() command. The simplest way to use the
* command is to specify two input arguments x + y, each one corresponding to one of the variables. The
* 18This is an oversimplification, but it’ll do for our purposes.
* - 141 -
* positive correlations negative correlations
* correlation example correlation example
* 0.0 0.0
* 0.33 ´0.33
* 0.66 ´0.66
* 1.0 ´1.0
* Figure 5.9: Illustration of the effect of varying the strength + direction of a correlation. In the left
* hand column, the correlations are 0, .33, .66 + 1. In the right hand column, the correlations are 0, -.33,
* -.66 + -1.
* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
* - 142 -
* following extract illustrates the basic usage of the function:19
* > cor( x = parenthood$dan.sleep, y = parenthood$dan.grump )
* [1] -0.903384
* However, the cor() function is a bit more powerful than this simple example suggests. For example, you
* can also calculate a complete “correlation matrix”, between all pairs of variables in the data frame:20
* > # correlate all pairs of variables in "parenthood":
* > cor( x = parenthood )
* dan.sleep baby.sleep dan.grump day
* dan.sleep 1.00000000 0.62794934 -0.90338404 -0.09840768
* baby.sleep 0.62794934 1.00000000 -0.56596373 -0.01043394
* dan.grump -0.90338404 -0.56596373 1.00000000 0.07647926
* day -0.09840768 -0.01043394 0.07647926 1.00000000
* 5.7.5 Interpreting a correlation
* Naturally, in real life you don’t see many correlations of 1. So how should you interpret a correlation
* of, say r “ .4? The honest answer is that it really depends on what you want to use the data for, +
* on how strong the correlations in your field tend to be. A friend of mine in engineering once argued that
* any correlation less than .95 is completely useless (I think he was exaggerating, even for engineering). On
* the other hand there are real cases – even in psychology – where you should really expect correlations
* that strong. For instance, one of the benchmark data sets used to test theories of how people judge
* similarities is so clean that any theory that can’t achieve a correlation of at least .9 really isn’t deemed to
* be successful. However, when looking for (say) elementary correlates of intelligence (e.g., inspection time,
* response time), if you get a correlation above .3 you’re doing very very well. In short, the interpretation
* of a correlation depends a lot on the context. That said, the rough guide in Table 5.2 is pretty typical.
* However, something that can never be stressed enough is that you should always look at the scatterplot
* before attaching any interpretation to the data. A correlation might not mean what you think it means.
* The classic illustration of this is “Anscombe’s Quartet” (Anscombe, 1973), which is a collection of four
* data sets. Each data set has two variables, an X + a Y . For all four data sets the mean value for X
* is 9 + the mean for Y is 7.5. The, SDs for all X variables are almost identical, as are
* those for the the Y variables. + in each case the correlation between X + Y is r “ 0.816. You can
* verify this yourself, since I happen to have saved the in a file21 called anscombesquartet.Rdata. The
* commands would be:
* > load( "anscombesquartet.Rdata" )
* > cor( X1, Y1 )
* [1] 0.8164205
* > cor( X2, Y2 )
* [1] 0.8162365
* + so on. You’d think that these four data setswould look pretty similar to one another. They do not.
* If we draw scatterplots of X against Y for all four variables, as shown in Figure 5.10 we see that all four
* 19If you are reading this after having already completed Chapter 11 you might be wondering about hypothesis tests for
* correlations. R has a function called cor.test() that runs a hypothesis test for a single correlation, + the psych package
* contains a version called corr.test() that can run tests for every correlation in a correlation matrix; hypothesis tests for
* correlations are discussed in more detail in Section 15.6.
* 20An alternative usage of cor() is to correlate one set of variables w/ another subset of variables. If X + Y are both
* data frames w/ the same number of rows, then cor(x = X, y = Y) will produce a correlation matrix that correlates all
* variables in X w/ all variables in Y.
* 21It’s also stored as a dataset that comes distributed w/ R. Just type anscombe at the command prompt.
* - 143 -
* Table 5.2: A rough guide to interpreting correlations. Note that I say a rough guide. There aren’t hard
* + fast rules for what counts as strong or weak relationships. It depends on the context.
* Correlation Strength Direction
* -1.0 to -0.9 Very strong Negative
* -0.9 to -0.7 Strong Negative
* -0.7 to -0.4 Moderate Negative
* -0.4 to -0.2 Weak Negative
* -0.2 to 0 Negligible Negative
* 0 to 0.2 Negligible Positive
* 0.2 to 0.4 Weak Positive
* 0.4 to 0.7 Moderate Positive
* 0.7 to 0.9 Strong Positive
* 0.9 to 1.0 Very strong Positive
* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
* of these are spectacularly different to each other. The lesson here, which so very many people seem to
* forget in real life is “always graph your raw data”. This will be the focus of Chapter 6.
* 5.7.6 Spearman’s rank correlations
* The Pearson correlation coefficient is useful for a lot of things, but it does have shortcomings. One
* issue in particular stands out: what it actually measures is the strength of the linear relationship between
* two variables. In other words, what it gives you is a measure of the extent to which the data all tend to
* fall on a single, perfectly straight line. Often, this is a pretty good approximation to what we mean when
* we say “relationship”, + so the Pearson correlation is a good thing to calculation. Sometimes, it isn’t.
* One very common situation where the Pearson correlation isn’t quite the right thing to use arises
* when an increase in one variable X really is reflected in an increase in another variable Y , but the nature
* of the relationship isn’t necessarily linear. An example of this might be the relationship between effort
* + reward when studying for an exam. If you put in zero effort (X) into learning a subject, then you
* should expect a grade of 0% (Y ). However, a little bit of effort will cause a massive improvement: just
* turning up to lectures means that you learn a fair bit, + if you just turn up to classes, + scribble a
* few things down so your grade might rise to 35%, all w/out a lot of effort. However, you just don’t get
* the same effect at the other end of the scale. As everyone knows, it takes a lot more effort to get a grade
* of 90% than it takes to get a grade of 55%. What this means is that, if I’ve got data looking at study
* effort + grades, there’s a pretty good chance that Pearson correlations will be misleading.
* To illustrate, consider the data plotted in Figure 5.11, showing the relationship between hours worked
* + grade received for 10 students taking some class. The curious thing about this – highly fictitious –
* data set is that increasing your effort always increases your grade. It might be by a lot or it might be by
* a little, but increasing effort will never decrease your grade. The data are stored in effort.Rdata:
* > load( "effort.Rdata" )
* > who(TRUE)
* -- Name -- -- Class -- -- Size --
* effort data.frame 10 x 2
* $hours numeric 10
* $grade numeric 10
* The raw data look like this:
* - 144 -
* 4 6 8 10 12 14
* 4 5 6 7 8 9 10
* X1
* Y1
* 4 6 8 10 12 14
* 3 4 5 6 7 8 9
* X2
* Y2
* 4 6 8 10 12 14
* 6 8 10 12
* X3
* Y3
* 8 10 12 14 16 18
* 6 8 10 12
* X4
* Y4
* Figure 5.10: Anscombe’s quartet. All four of these data sets have a Pearson correlation of r “ .816, but
* they are qualitatively different from one another.
* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
* > effort
* hours grade
* 1 2 13
* 2 76 91
* 3 40 79
* 4 6 14
* 5 16 21
* 6 28 74
* 7 27 47
* 8 59 85
* 9 46 84
* 10 68 88
* If we run a standard Pearson correlation, it shows a strong relationship between hours worked + grade
* received,
* - 145 -
* 0 20 40 60 80
* 0 20 40 60 80 100
* Hours Worked
* Grade Received
* Figure 5.11: The relationship between hours worked + grade received, for a toy data set consisting of
* only 10 students (each circle corresponds to one student). The dashed line through the middle shows
* the linear relationship between the two variables. This produces a strong Pearson correlation of r “ .91.
* However, the interesting thing to note here is that there’s actually a perfect monotonic relationship
* between the two variables: in this toy example at least, increasing the hours worked always increases
* the grade received, as illustrated by the solid line. This is reflected in a Spearman correlation of ρ “ 1.
* W/ such a small data set, however, it’s an open question as to which version better describes the actual
* relationship involved.
* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
* > cor( effort$hours, effort$grade )
* [1] 0.909402
* but this doesn’t actually capture the observation that increasing hours worked always increases the grade.
* There’s a sense here in which we want to be able to say that the correlation is perfect but for a somewhat
* different notion of what a “relationship” is. What we’re looking for is something that captures the fact
* that there is a perfect ordinal relationship here. That is, if student 1 works more hours than student 2,
* then we can guarantee that student 1 will get the better grade. That’s not what a correlation of r “ .91
* says at all.
* How should we address this? Actually, it’s really easy: if we’re looking for ordinal relationships, all
* we have to do is treat the data as if it were ordinal scale! So, instead of measuring effort in terms of
* “hours worked”, lets rank all 10 of our students in order of hours worked. That is, student 1 did the least
* work out of anyone (2 hours) so they get the lowest rank (rank = 1). Student 4 was the next laziest,
* putting in only 6 hours of work in over the whole semester, so they get the next lowest rank (rank = 2).
* Notice that I’m using “rank =1” to mean “low rank”. Sometimes in everyday language we talk about
* “rank = 1” to mean “top rank” rather than “bottom rank”. So be careful: you can rank “from smallest
* value to largest value” (i.e., small equals rank 1) or you can rank “from largest value to smallest value”
* (i.e., large equals rank 1). In this case, I’m ranking from smallest to largest, b/c that’s the default
* way that R does it. But in real life, it’s really easy to forget which way you set things up, so you have to
* put a bit of effort into remembering!
* - 146 -
* Okay, so let’s have a look at our students when we rank them from worst to best in terms of effort
* + reward:
* rank (hours worked) rank (grade received)
* student 1 1 1
* student 2 10 10
* student 3 6 6
* student 4 2 2
* student 5 3 3
* student 6 5 5
* student 7 4 4
* student 8 8 8
* student 9 7 7
* student 10 9 9
* Hm. These are identical. The student who put in the most effort got the best grade, the student w/ the
* least effort got the worst grade, etc. We can get R to construct these rankings using the rank() function,
* like this:
* > hours.rank <- rank( effort$hours ) # rank students by hours worked
* > grade.rank <- rank( effort$grade ) # rank students by grade received
* As the table above shows, these two rankings are identical, so if we now correlate them we get a perfect
* relationship:
* > cor( hours.rank, grade.rank )
* [1] 1
* What we’ve just re-invented is Spearman’s rank order correlation, usually denoted ρ to distinguish
* it from the Pearson correlation r. We can calculate Spearman’s ρ using R in two different ways. Firstly
* we could do it the way I just showed, using the rank() function to construct the rankings, + then
* calculate the Pearson correlation on these ranks. However, that’s way too much effort to do every time.
* It’s much easier to just specify the method argument of the cor() function.
* > cor( effort$hours, effort$grade, method = "spearman")
* [1] 1
* The default value of the method argument is "pearson", which is why we didn’t have to specify it earlier
* on when we were doing Pearson correlations.
* 5.7.7 The ❝♦rr❡❧❛t❡✭✮ function
* As we’ve seen, the cor() function works pretty well, + handles many of the situations that you
* might be interested in. One thing that many beginners find frustrating, however, is the fact that it’s not
* built to handle non-numeric variables. From a statistical perspective, this is perfectly sensible: Pearson
* + Spearman correlations are only designed to work for numeric variables, so the cor() function spits
* out an error.
* Here’s what I mean. Suppose you were keeping track of how many hours you worked in any given
* day, + counted how many tasks you completed. If you were doing the tasks for money, you might also
* want to keep track of how much pay you got for each job. It would also be sensible to keep track of the
* - 147 -
* weekday on which you actually did the work: most of us don’t work as much on Saturdays or Sundays. If
* you did this for 7 weeks, you might end up w/ a data set that looks like this one:
* > load("work.Rdata")
* > who(TRUE)
* -- Name -- -- Class -- -- Size --
* work data.frame 49 x 7
* $hours numeric 49
* $tasks numeric 49
* $pay numeric 49
* $day integer 49
* $weekday factor 49
* $week numeric 49
* $day.type factor 49
* > head(work)
* hours tasks pay day weekday week day.type
* 1 7.2 14 41 1 Tuesday 1 weekday
* 2 7.4 11 39 2 Wednesday 1 weekday
* 3 6.6 14 13 3 Thursday 1 weekday
* 4 6.5 22 47 4 Friday 1 weekday
* 5 3.1 5 4 5 Saturday 1 weekend
* 6 3.0 7 12 6 Sunday 1 weekend
* Obviously, I’d like to know something about how all these variables correlate w/ one another. I could
* correlate hours w/ pay quite using cor(), like so:
* > cor(work$hours,work$pay)
* [1] 0.7604283
* But what if I wanted a quick + easy way to calculate all pairwise correlations between the numeric
* variables? I can’t just input the work data frame, b/c it contains two factor variables, weekday +
* day.type. If I try this, I get an error:
* > cor(work)
* Error in cor(work) : ’x’ must be numeric
* It order to get the correlations that I want using the cor() function, is create a new data frame that
* doesn’t contain the factor variables, + then feed that new data frame into the cor() function. It’s not
* actually very hard to do that, + I’ll talk about how to do it properly in Section 7.5. But it would be
* nice to have some function that is smart enough to just ignore the factor variables. That’s where the
* correlate() function in the lsr package can be handy. If you feed it a data frame that contains factors,
* it knows to ignore them, + returns the pairwise correlations only between the numeric variables:
* > correlate(work)
* CORRELATIONS
* ============
* - correlation type: pearson
* - correlations shown only when both variables are numeric
* hours tasks pay day weekday week day.type
* hours . 0.800 0.760 -0.049 . 0.018 .
* tasks 0.800 . 0.720 -0.072 . -0.013 .
* - 148 -
* pay 0.760 0.720 . 0.137 . 0.196 .
* day -0.049 -0.072 0.137 . . 0.990 .
* weekday . . . . . . .
* week 0.018 -0.013 0.196 0.990 . . .
* day.type . . . . . . .
* The output here shows a . whenever one of the variables is non-numeric. It also shows a . whenever a
* variable is correlated w/ itself (it’s not a meaningful thing to do). The correlate() function can also
* do Spearman correlations, by specifying the corr.method to use:
* > correlate( work, corr.method="spearman" )
* CORRELATIONS
* ============
* - correlation type: spearman
* - correlations shown only when both variables are numeric
* hours tasks pay day weekday week day.type
* hours . 0.805 0.745 -0.047 . 0.010 .
* tasks 0.805 . 0.730 -0.068 . -0.008 .
* pay 0.745 0.730 . 0.094 . 0.154 .
* day -0.047 -0.068 0.094 . . 0.990 .
* weekday . . . . . . .
* week 0.010 -0.008 0.154 0.990 . . .
* day.type . . . . . . .
* Obviously, there’s no new functionality in the correlate() function, + any advanced R user would be
* perfectly capable of using the cor() function to get these numbers out. But if you’re not yet comfortable
* w/ extracting a subset of a data frame, the correlate() function is for you.
* 5.8
* Handling missing values
* There’s one last topic that I want to discuss briefly in this chapter, + that’s the issue of missing
* data. Real data sets very frequently turn out to have missing values: perhaps someone forgot to fill in
* a particular survey question, for instance. Missing data can be the source of a lot of tricky issues, most
* of which I’m going to gloss over. However, at a minimum, you need to understand the basics of handling
* missing data in R.
* 5.8.1 The single variable case
* Let’s start w/ the simplest case, in which you’re trying to calculate descriptive statistics for a single
* variable which has missing data. In R, this means that there will be NA values in your data vector. Let’s
* create a variable like that:
* > partial <- c(10, 20, NA, 30)
* Let’s assume that you want to calculate the mean of this variable. By default, R assumes that you want
* to calculate the mean using all four elements of this vector, which is probably the safest thing for a dumb
* - 149 -
* automaton to do, but it’s rarely what you actually want. Why not? Well, remember that the basic
* interpretation of NA is “I don’t know what this number is”. This means that 1 + NA = NA: if I add 1 to
* some number that I don’t know (i.e., the NA) then the answer is also a number that I don’t know. As
* a consequence, if you don’t explicitly tell R to ignore the NA values, + the data set does have missing
* values, then the output will itself be a missing value. If I try to calculate the mean of the partial vector,
* w/out doing anything about the missing value, here’s what happens:
* > mean( x = partial )
* [1] NA
* Technically correct, but deeply unhelpful.
* To fix this, all of the descriptive statistics functions that I’ve discussed in this chapter (w/ the
* exception of cor() which is a special case I’ll discuss below) have an optional argument called na.rm,
* which is shorthand for “remove NA values”. By default, na.rm = FALSE, so R does nothing about the
* missing data problem. Let’s try setting na.rm = TRUE + see what happens:
* When calculating sums + means when missing data are present (i.e., when there are NA values)
* there’s actually an additional argument to the function that you should be aware of. This argument is
* called na.rm, + is a logical value indicating whether R should ignore (or “remove”) the missing data for
* the purposes of doing the calculations. By default, R assumes that you want to keep the missing values,
* so unless you say otherwise it will set na.rm = FALSE. However, R assumes that 1 + NA = NA: if I add 1 to
* some number that I don’t know (i.e., the NA) then the answer is also a number that I don’t know. As
* a consequence, if you don’t explicitly tell R to ignore the NA values, + the data set does have missing
* values, then the output will itself be a missing value. This is illustrated in the following extract:
* > mean( x = partial, na.rm = TRUE )
* [1] 20
* Notice that the mean is 20 (i.e., 60 / 3) + not 15. When R ignores a NA value, it genuinely ignores it. In
* effect, the calculation above is identical to what you’d get if you asked for the mean of the three-element
* vector c(10, 20, 30).
* As indicated above, this isn’t unique to the mean() function. Pretty much all of the other functions
* that I’ve talked about in this chapter have an na.rm argument that indicates whether it should ignore
* missing values. However, its behaviour is the same for all these functions, so I won’t waste everyone’s
* time by demonstrating it separately for each one.
* 5.8.2 Missing values in pairwise calculations
* I mentioned earlier that the cor() function is a special case. It doesn’t have an na.rm argument,
* b/c the story becomes a lot more complicated when more than one variable is involved. What it
* does have is an argument called use which does roughly the same thing, but you need to think little
* more carefully about what you want this time. To illustrate the issues, let’s open up a data set that has
* missing values, parenthood2.Rdata. This file contains the same data as the original parenthood data,
* but w/ some values deleted. It contains a single data frame, parenthood2:
* > load( "parenthood2.Rdata" )
* > print( parenthood2 )
* dan.sleep baby.sleep dan.grump day
* 1 7.59 NA 56 1
* 2 7.91 11.66 60 2
* 3 5.14 7.92 82 3
* 4 7.71 9.61 55 4
* - 150 -
* 5 6.68 9.75 NA 5
* 6 5.99 5.04 72 6
* BLAH BLAH BLAH
* If I calculate my descriptive statistics using the describe() function
* > describe( parenthood2 )
* var n mean sd median trimmed mad min max BLAH
* dan.sleep 1 91 6.98 1.02 7.03 7.02 1.13 4.84 9.00 BLAH
* baby.sleep 2 89 8.11 2.05 8.20 8.13 2.28 3.25 12.07 BLAH
* dan.grump 3 92 63.15 9.85 61.00 62.66 10.38 41.00 89.00 BLAH
* day 4 100 50.50 29.01 50.50 50.50 37.06 1.00 100.00 BLAH
* we can see from the n column that there are 9 missing values for dan.sleep, 11 missing values for
* baby.sleep + 8 missing values for dan.grump.
* 22 Suppose what I would like is a correlation matrix. +
* let’s also suppose that I don’t bother to tell R how to handle those missing values. Here’s what happens:
* > cor( parenthood2 )
* dan.sleep baby.sleep dan.grump day
* dan.sleep 1 NA NA NA
* baby.sleep NA 1 NA NA
* dan.grump NA NA 1 NA
* day NA NA NA 1
* Annoying, but it kind of makes sense. If I don’t know what some of the values of dan.sleep + baby.sleep
* actually are, then I can’t possibly know what the correlation between these two variables is either, since
* the formula for the correlation coefficient makes use of every single observation in the data set. Once
* again, it makes sense: it’s just not particularly helpful.
* To make R behave more sensibly in this situation, you need to specify the use argument to the
* cor() function. There are several different values that you can specify for this, but the two that we
* care most about in practice tend to be "complete.obs" + "pairwise.complete.obs". If we specify use
* = "complete.obs", R will completely ignore all cases (i.e., all rows in our parenthood2 data frame) that
* have any missing values at all. So, for instance, if you look back at the extract earlier when I used the
* head() function, notice that observation 1 (i.e., day 1) of the parenthood2 data set is missing the value for
* baby.sleep, but is otherwise complete? Well, if you choose use = "complete.obs" R will ignore that row
* completely: that is, even when it’s trying to calculate the correlation between dan.sleep + dan.grump,
* observation 1 will be ignored, b/c the value of baby.sleep is missing for that observation. Here’s
* what we get:
* > cor(parenthood2, use = "complete.obs")
* dan.sleep baby.sleep dan.grump day
* dan.sleep 1.00000000 0.6394985 -0.89951468 0.06132891
* baby.sleep 0.63949845 1.0000000 -0.58656066 0.14555814
* dan.grump -0.89951468 -0.5865607 1.00000000 -0.06816586
* day 0.06132891 0.1455581 -0.06816586 1.00000000
* The other possibility that we care about, + the one that tends to get used more often in practice, is
* to set use = "pairwise.complete.obs". When we do that, R only looks at the variables that it’s trying to
* correlate when determining what to drop. So, for instance, since the only missing value for observation
* 1 of parenthood2 is for baby.sleep R will only drop observation 1 when baby.sleep is one of the variables
* involved: + so R keeps observation 1 when trying to correlate dan.sleep + dan.grump. When we do
* it this way, here’s what we get:
* 22It’s worth noting that, even though we have missing data for each of these variables, the output doesn’t contain any NA
* values. This is b/c, while describe() also has an na.rm argument, the default value for this function is na.rm = TRUE.
* - 151 -
* > cor(parenthood2, use = "pairwise.complete.obs")
* dan.sleep baby.sleep dan.grump day
* dan.sleep 1.00000000 0.61472303 -0.903442442 -0.076796665
* baby.sleep 0.61472303 1.00000000 -0.567802669 0.058309485
* dan.grump -0.90344244 -0.56780267 1.000000000 0.005833399
* day -0.07679667 0.05830949 0.005833399 1.000000000
* Similar, but not quite the same. It’s also worth noting that the correlate() function (in the lsr package)
* automatically uses the “pairwise complete” method:
* > correlate(parenthood2)
* CORRELATIONS
* ============
* - correlation type: pearson
* - correlations shown only when both variables are numeric
* dan.sleep baby.sleep dan.grump day
* dan.sleep . 0.615 -0.903 -0.077
* baby.sleep 0.615 . -0.568 0.058
* dan.grump -0.903 -0.568 . 0.006
* day -0.077 0.058 0.006 .
* The two approaches have different strengths + weaknesses. The “pairwise complete” approach has
* the advantage that it keeps more observations, so you’re making use of more of your data + (as we’ll
* discuss in tedious detail in Chapter 10) + it improves the reliability of your estimated correlation.
* On the other hand, it means that every correlation in your correlation matrix is being computed from
* a slightly different set of observations, which can be awkward when you want to compare the different
* correlations that you’ve got.
* So which method should you use? It depends a lot on why you think your values are missing, +
* probably depends a little on how paranoid you are. For instance, if you think that the missing values
* were “chosen” completely randomly23 then you’ll probably want to use the pairwise method. If you think
* that missing data are a cue to thinking that the whole observation might be rubbish (e.g., someone just
* selecting arbitrary responses in your questionnaire), but that there’s no pattern to which observations are
* “rubbish” then it’s probably safer to keep only those observations that are complete. If you think there’s
* something systematic going on, in that some observations are more likely to be missing than others, then
* you have a much trickier problem to solve, + one that is beyond the scope of this book.