ACED Data

About ACED

ACED stands for Adaptive Content and Evidence-based Diagnosis. It was an intelligent tutoring system built by Val Shute and others [@FatHog,@shute2006]. It's domain was algebraic sequences, although only the geometric sequences branch was involved in the field trials.

ACED Tasks and Feedback

ACED uses extended constructed response items or tasks.

For each item, if the student got the item wrong they could be provided with elaborated feedback.

Bayesian Network Scoring Engine

As the students are solving problems, they're abilities can be measured as the network goes on. The scoring model is a Bayesian network [@bninea]. The specific Bayesian network used in ACED is shown below.

In this Bayesian network, each node in the graph corresponds to a (latent) variable which takes on one of the values *high*, *medium* or *low*. The Bayes net produces a probability distribution over the latent variable.

The columns `P.sgp..H`, `P.sgp..M` and `P.sgp..L` give the probabilities of the that the student is in the *high*, *medium* or *low* state. The expected value, `EAP.sgp` (expected a posteriori, solve geometric problems) is computed by assigning *high = 1*, *medium = 0* and *low = -1*, and then taking the expected value, `EAP.sgp <- 1*P.sgp..H + 0*P.sgp..M + -1*P.sgp..L`.

The other EAP variables represent the other variables in the model.

One advantage of using a model like a Bayesian network is that it can estimate the student's ability using only part of the data. It can also calculate for any item its *expected weight of*

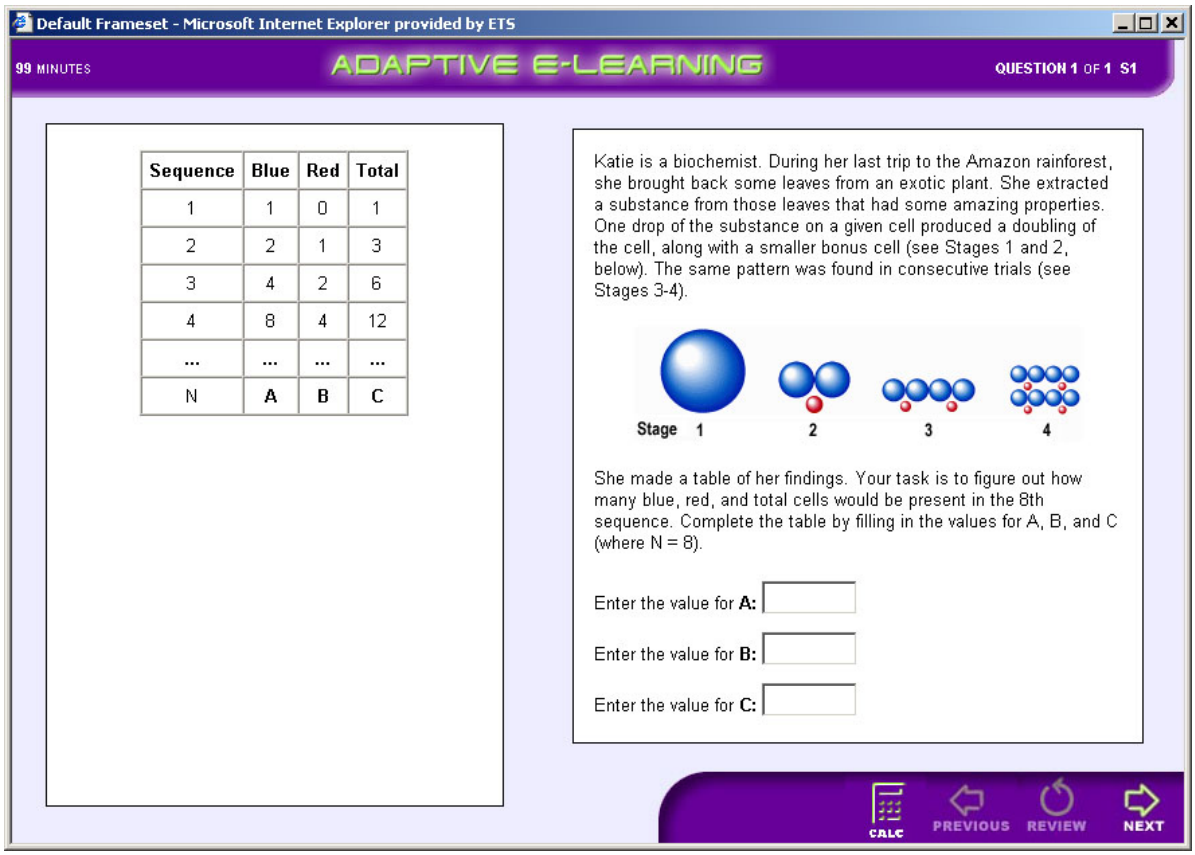


Figure 1: Sample ACED Item

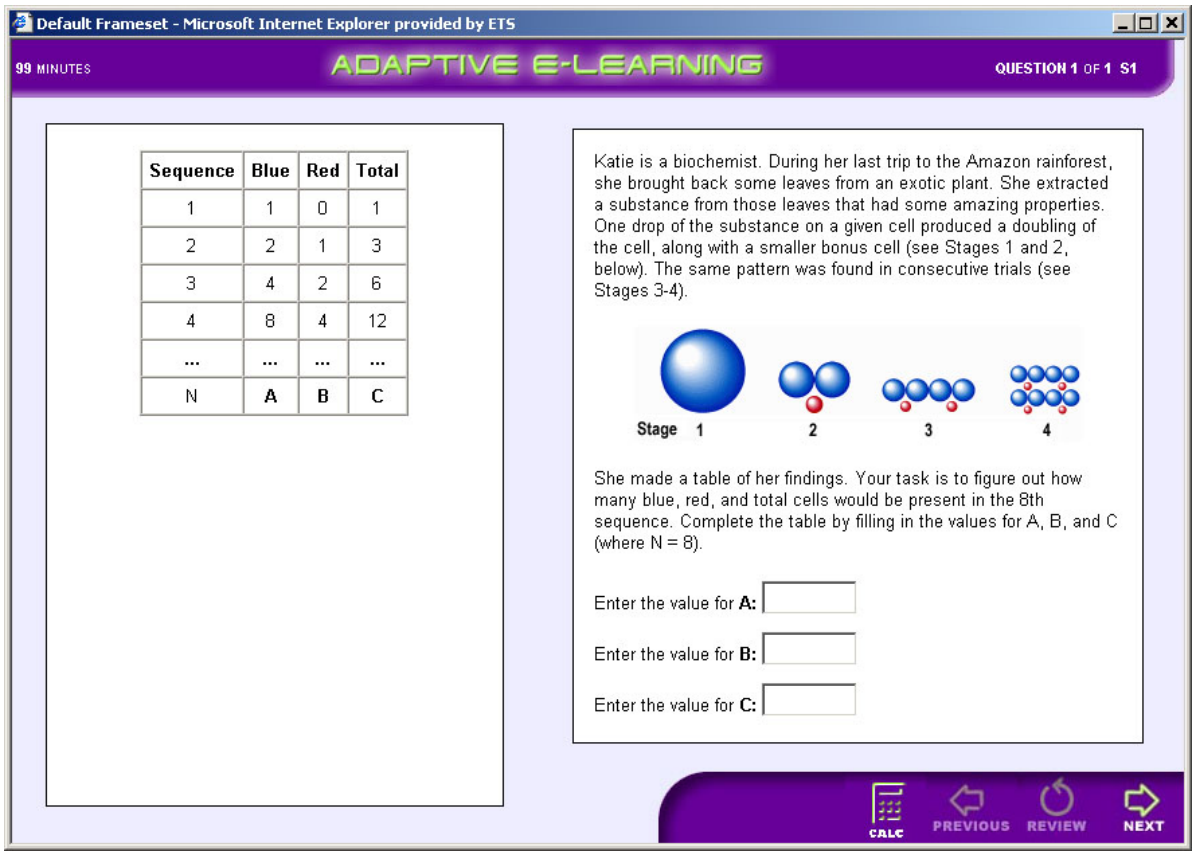


Figure 2: Sample Item Feedback

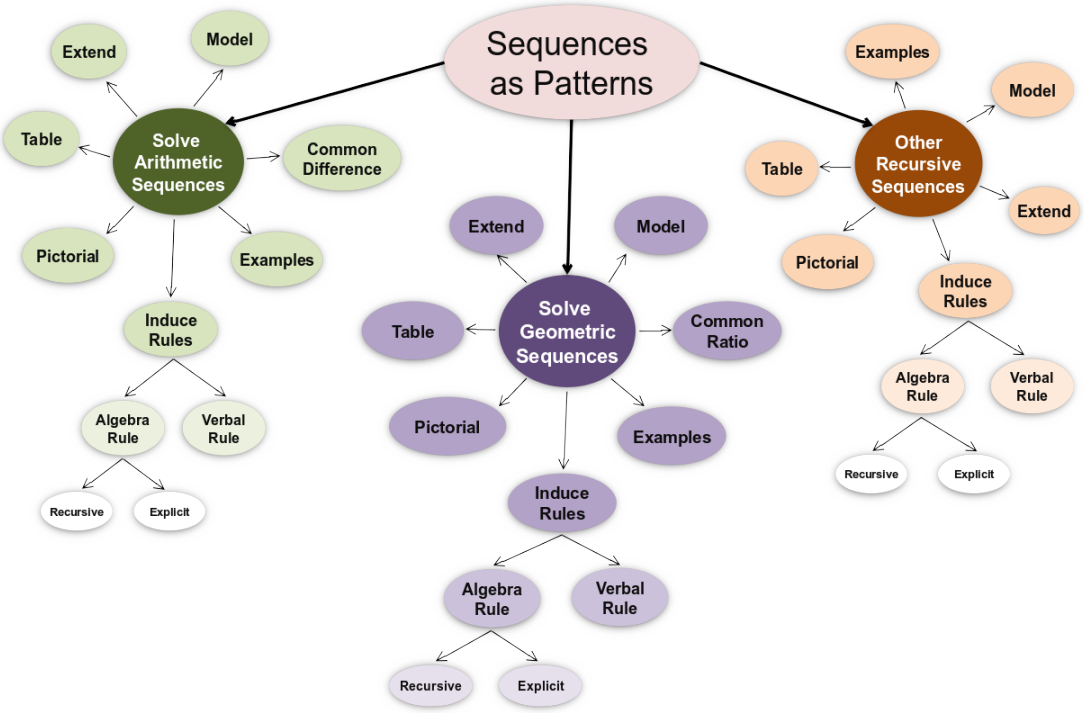


Figure 3: ACED Competency Model

evidence [@bninea], the predicted amount of information the item could provide about the target node in the network (*Solve Geometric Problems*).

Pretest and posttest

In addition to the ACED items, there is also a 25 item pretest and post-test. More about this in another case study.

The experiment

Around 300 (slightly less, after restricting the sample to the subjects for which consent and assent were obtained) middle school students participated in the study. Arithmetic sequences were part of the normal math curriculum, but geometric sequences were not.

The students were split into four groups:

A. Full feedback/adaptive sequence. The full feedback was turned on and the expected weight of evidence algorithm is used to make a custom sequence for each student.

B. Accuracy-only feedback/adaptive sequence. Instead of the full feedback, the student is only told that their solution is correct or incorrect. The same adaptive sequence algorithm is used.

C. Full feedback/linear sequence. The full feedback was turned on, but the same fixed (linear) sequence of items is used for all students in this group.

D. Control. These students took the pretest and post-test, but did independent study while other students were using ACED.

This is an incomplete factorial design (i.e., we have two factors, but the accuracy only/linear sequence condition is replaced with the control).

The research questions

1. Do the pretest, posttest and internal game measures measure the same thing? (Validity and Reliability)
2. Does using ACED increase students understanding of algebraic sequences?
3. Do feedback and/or adaptive sequencing affect the amount of learning?

The first can be answered with simple regression. The last two can be answered with a technique called the analysis of covariance (ANCOVA).

Loading R packages

R functions (and data) are bundled together in *packages*. Various statisticians have produced packages to extend the abilities of R. A complete list of these packages (there are a whole lot) can be found on CRAN.

We will use two: `DescTools` and `tidyverse`.

`DescTools` has a bunch of tools for descriptive statistics. The meta-package `tidyverse` loads a bunch of tools which make R syntax a little bit closer to how statisticians think about data. @r4ds2e explores these in detail.

Installing packages

R comes with a bunch of core packages which perform many analyses. Additional packages need to be downloaded from the repository (CRAN, one of its mirrors, or a non-CRAN repository like Bioconductor or R-Universe). Copying the package files from the repository to the local machine is called installation. This only needs to be done once, but packages might need to be updated if a new version of the package (or R) is released.

There are two ways to install packages: using RStudio and using R.

Using RStudio, select `Tools > Install Packages ...` and put in the name of the packages you want to install.

The second way is to call the `install.packages` function in R.

```
## Skip this step if packages are already installed.
if (length(find.package(c("tidyverse","DescTools"))) < 2L) {
  ## Set the repos field to avoid having R prompt you to select a mirror.
  install.packages(c("tidyverse","DescTools"),
                   repos="https://cloud.r-project.org/")
}
```

Using the package

Once the package is installed, it can be used. First, we can specify that we want the function from a specific package by putting the package name and `::` in front of the function name. For example, `DescTools::Cor()` refers to the function `Cor()` in the `DescTools` package, while `stats::cor()` is the similar function in the core `stats` package.

If we are going to use the package a lot, we can attach it the list of packages R searches for functions, by using the function `library()`. The function `search()` shows which packages are currently attached.

```
search()
```

```
[1] ".GlobalEnv"        "package:stats"     "package:graphics"
[4] "package:grDevices" "package:utils"     "package:datasets"
[7] "package:methods"   "Autoloads"         "package:base"
```

Calling `library()` adds new packages to the search list.

```
library(DescTools)
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.2     v readr     2.1.4
v forcats   1.0.0     v stringr   1.5.0
v ggplot2   3.4.3     v tibble    3.2.1
v lubridate 1.9.2     v tidyr     1.3.0
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to beco
```

```
search()
```

```
 [1] ".GlobalEnv"        "package:lubridate" "package:forcats"
 [4] "package:stringr"   "package:dplyr"     "package:purrr"
 [7] "package:readr"     "package:tidyr"     "package:tibble"
[10] "package:ggplot2"   "package:tidyverse" "package:DescTools"
[13] "package:stats"     "package:graphics"  "package:grDevices"
[16] "package:utils"     "package:datasets"  "package:methods"
[19] "Autoloads"         "package:base"
```

It is very common to have a couple of calls to `library()` at the start of each analysis script.

Getting help on functions and packages.

The function `help(`*functionname*`)` will give information about a function. This is abbreviated `?`*functionname*.

```
help("Cor")
```

Note that in R Studio, there is a tab in which help it appears. Note that at the bottom of the help is an example of the command in action, which you can run to see how it works.

You can get help on the whole package by calling `help(package="XXX")`.

```
help(package="DescTools")
```

Finally, you should credit the package authors. You can find the right way to do this using the `citation()` function.

```
citation()
```

```
To cite R in publications use:

  R Core Team (2023). _R: A Language and Environment for Statistical
  Computing_. R Foundation for Statistical Computing, Vienna, Austria.
  <https://www.R-project.org/>.

A BibTeX entry for LaTeX users is

  @Manual{,
    title = {R: A Language and Environment for Statistical Computing},
    author = {{R Core Team}},
    organization = {R Foundation for Statistical Computing},
    address = {Vienna, Austria},
    year = {2023},
    url = {https://www.R-project.org/},
  }

We have invested a lot of time and effort in creating R, please cite it
when using it for data analysis. See also 'citation("pkgname")' for
citing R packages.
```

```
citation("DescTools")
```

```
To cite package 'DescTools' in publications use:

  Signorell A (2023). _DescTools: Tools for Descriptive Statistics_. R
  package version 0.99.49,
  <https://CRAN.R-project.org/package=DescTools>.

A BibTeX entry for LaTeX users is

  @Manual{,
    title = {DescTools: Tools for Descriptive Statistics},
    author = {Andri Signorell},
    year = {2023},
    note = {R package version 0.99.49},
    url = {https://CRAN.R-project.org/package=DescTools},
  }
```

```
citation("tidyverse")
```

```
To cite package 'tidyverse' in publications use:

  Wickham H, Averick M, Bryan J, Chang W, McGowan LD, François R,
  Grolemund G, Hayes A, Henry L, Hester J, Kuhn M, Pedersen TL, Miller
  E, Bache SM, Müller K, Ooms J, Robinson D, Seidel DP, Spinu V,
  Takahashi K, Vaughan D, Wilke C, Woo K, Yutani H (2019). "Welcome to
  the tidyverse." _Journal of Open Source Software_, *4*(43), 1686.
  doi:10.21105/joss.01686 <https://doi.org/10.21105/joss.01686>.

A BibTeX entry for LaTeX users is

  @Article{,
    title = {Welcome to the {tidyverse}},
    author = {Hadley Wickham and Mara Averick and Jennifer Bryan and Winston Chang and Lucy
    year = {2019},
    journal = {Journal of Open Source Software},
    volume = {4},
    number = {43},
    pages = {1686},
    doi = {10.21105/joss.01686},
  }
```

Loading the Data

Importing the data

The functions `base::read.csv()` and `readr::read_csv()` will read a a comma separated value (csv) file.

The function `summary()` shows a description of the variables.

The function `head()` shows the first couple of lines.

```
ACEDextract <- read_csv("ACED_extract1.csv")
```

```
Rows: 290 Columns: 29
-- Column specification --------------------------------------------------------
Delimiter: ","
chr  (7): SubjID, Session, Cond_code, Sequencing, Feedback, Gender, Level_Code
dbl (22): Correct, Incorrect, Reamaining, ElapsedTime, Race, pre_scaled, pos...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Viewing the data

A couple of ways to view the data.

`summary()` gives a summary.

```
summary(ACEDextract)
```

```
   SubjID            Session           Cond_code          Sequencing
 Length:290         Length:290         Length:290         Length:290
 Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character


  Feedback            Correct          Incorrect         Reamaining
 Length:290         Min.   :-999.0   Min.   :-999.00   Min.   :-999.0
 Class :character   1st Qu.:   8.0   1st Qu.:  18.00   1st Qu.:   0.0
 Mode  :character   Median :  18.0   Median :  34.00   Median :   0.0
```

```
                  Mean   :-188.9   Mean   :-177.24   Mean   :-204.1
                  3rd Qu.:  27.0   3rd Qu.:  44.75   3rd Qu.:   0.0
                  Max.   :  51.0   Max.   :  59.00   Max.   :  43.0
 ElapsedTime        Gender              Race           Level_Code
Min.   :-999.0   Length:290         Min.   :-999.000   Length:290
1st Qu.:-999.0   Class :character   1st Qu.:   3.000   Class :character
Median :1506.0   Mode  :character   Median :   6.000   Mode  :character
Mean   : 896.7                      Mean   :  -4.807
3rd Qu.:2093.2                      3rd Qu.:   7.000
Max.   :2693.0                      Max.   :   8.000
  pre_scaled       post_scaled        Form_Order       EAP.sgp
Min.   :-999.00   Min.   :-999.00   Min.   :1.000   Min.   :-999.0000
1st Qu.:  44.00   1st Qu.:  47.00   1st Qu.:1.000   1st Qu.:   0.0000
Median :  50.00   Median :  54.00   Median :1.000   Median :   0.0155
Mean   :  42.72   Mean   :  47.31   Mean   :1.486   Mean   :-206.2563
3rd Qu.:  57.00   3rd Qu.:  61.00   3rd Qu.:2.000   3rd Qu.:   0.7760
Max.   :  78.00   Max.   :  84.00   Max.   :2.000   Max.   :   1.9980
    EAP.cr            EAP.dt            EAP.eg            EAP.exp
Min.   :-999.000   Min.   :-999.0000   Min.   :-999.000   Min.   :-999.000
1st Qu.:   0.439   1st Qu.:   0.4220   1st Qu.:   0.010   1st Qu.:   0.003
Median :   1.008   Median :   0.4270   Median :   0.017   Median :   0.014
Mean   :-205.678   Mean   :-206.2295   Mean   :-206.349   Mean   :-206.612
3rd Qu.:   1.822   3rd Qu.:   0.6698   3rd Qu.:   0.217   3rd Qu.:   0.045
Max.   :   2.000   Max.   :   0.9520   Max.   :   1.935   Max.   :   1.593
   EAP.ext            EAP.mod            EAP.rr
Min.   :-999.0000   Min.   :-999.0000   Min.   :-999.0000
1st Qu.:   0.1628   1st Qu.:   0.0102   1st Qu.:   0.1040
Median :   1.1435   Median :   0.0545   Median :   0.4985
Mean   :-205.6309   Mean   :-206.3877   Mean   :-206.1130
3rd Qu.:   1.9163   3rd Qu.:   0.4305   3rd Qu.:   0.8668
Max.   :   2.0000   Max.   :   1.6970   Max.   :   1.9740
   EAP.tab            EAP.vr            EAP.pic            P.sgp..H
Min.   :-999.0000   Min.   :-999.0000   Min.   :-999.000   Min.   :-999.0000
1st Qu.:   0.0462   1st Qu.:   0.0540   1st Qu.:   0.019   1st Qu.:   0.0000
Median :   0.2780   Median :   0.2060   Median :   0.047   Median :   0.0000
Mean   :-206.1436   Mean   :-206.3237   Mean   :-206.443   Mean   :-206.5617
3rd Qu.:   0.9348   3rd Qu.:   0.5235   3rd Qu.:   0.241   3rd Qu.:   0.0318
Max.   :   1.9740   Max.   :   1.7980   Max.   :   1.895   Max.   :   0.9980
   P.sgp..M           P.sgp..L
Min.   :-999.0000   Min.   :-999.000
1st Qu.:   0.0000   1st Qu.:   0.000
Median :   0.0130   Median :   0.518
Mean   :-206.5124   Mean   :-206.202
```

```
 3rd Qu.:   0.2725   3rd Qu.:   0.993
 Max.   :   0.9030   Max.   :   1.000
```

`head()` shows the first couple of lines.

```
head(ACEDextract)
```

```
# A tibble: 6 x 29
  SubjID Session Cond_code     Sequencing Feedback Correct Incorrect Reamaining
  <chr>  <chr>   <chr>         <chr>      <chr>      <dbl>     <dbl>      <dbl>
1 S052   c01     adaptive_acc  Adaptive   Accuracy      20        14         29
2 S053   c01     adaptive_full Adaptive   Full          12        51          0
3 S054   c01     linear_full   Linear     Full          34        29          0
4 S055   c01     adaptive_acc  Adaptive   Accuracy       8        55          0
5 S056   c01     adaptive_full Adaptive   Full          20        43          0
6 S057   c01     linear_full   Linear     Full          25        28         10
# i 21 more variables: ElapsedTime <dbl>, Gender <chr>, Race <dbl>,
#   Level_Code <chr>, pre_scaled <dbl>, post_scaled <dbl>, Form_Order <dbl>,
#   EAP.sgp <dbl>, EAP.cr <dbl>, EAP.dt <dbl>, EAP.eg <dbl>, EAP.exp <dbl>,
#   EAP.ext <dbl>, EAP.mod <dbl>, EAP.rr <dbl>, EAP.tab <dbl>, EAP.vr <dbl>,
#   EAP.pic <dbl>, P.sgp..H <dbl>, P.sgp..M <dbl>, P.sgp..L <dbl>
```

In RStudio, you can also use `view()` to open a viewer window, or find the variable name in the environment tab.

Just typing the name of the variable will dump it all out. That is usually a mistake if there are a lot of cases. But in RStudio, it give you a nice interactive browser.

```
ACEDextract
```

A quick note on subsetting

We can use the expression `X[i,j]` to extract a single value.

Can also use vectors to get subsets.

```
ACEDextract[1,1]
```

```
# A tibble: 1 x 1
  SubjID
  <chr>
1 S052
```

```
ACEDextract[1:10,c("SubjID","Cond_code")]
```

```
# A tibble: 10 x 2
   SubjID Cond_code
   <chr>  <chr>
 1 S052   adaptive_acc
 2 S053   adaptive_full
 3 S054   linear_full
 4 S055   adaptive_acc
 5 S056   adaptive_full
 6 S057   linear_full
 7 S058   adaptive_acc
 8 S059   adaptive_full
 9 S060   linear_full
10 S061   adaptive_acc
```

Tibbles and data.frames behave somewhat differently here. Selecting a single value from a data frame gives a scalar (actually, vector of length 1), but from a tibble, gives a tibble of size 1 by 1. Use `as.numeric()` or `as.character()` to convert to a vector.

```
as.character(ACEDextract[1,1])
```

```
[1] "S052"
```

Selecting variables

Variables can be extracted by leaving the rows blank. A single variable can be extracted using the `$` operator.

```
summary(ACEDextract$Correct)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -999.0     8.0    18.0  -188.9    27.0    51.0
```

It can also be done using the tidyverse `select` (multiple variables) of `pull` (single variable).

```
  select(ACEDextract,starts_with("P.sgp"))
```

```
# A tibble: 290 x 3
   P.sgp..H P.sgp..M P.sgp..L
      <dbl>    <dbl>    <dbl>
 1    0.747    0.252    0.001
 2    0        0.001    0.999
 3    0.69     0.307    0.003
 4    0        0        1
 5    0        0.026    0.974
 6    0.258    0.622    0.119
 7    0.005    0.562    0.433
 8    0        0.053    0.947
 9    0.049    0.903    0.048
10    0.001    0.096    0.903
# i 280 more rows
```

The output of `select` is a tibble, even if it is just a single variable. If we want a vector instead, use `pull`.

```
  head(pull(ACEDextract,EAP.sgp))
```

```
[1] 1.747 0.001 1.687 0.000 0.026 1.139
```

Selecting cases

Frequently, we want to identify a subset of cases based on some logical condition.

We can use a logical subscript to do this, use a logical expression which evaluates to true of false for each row of the data frame or tibble.

```
  head(ACEDextract$Cond_code!="control")
```

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

```
  summary(ACEDextract[ACEDextract$Cond_code!="control",1:5])
```

```
   SubjID             Session            Cond_code          Sequencing
 Length:230         Length:230         Length:230         Length:230
 Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character
   Feedback
 Length:230
 Class :character
 Mode  :character
```

We can also use the `filter()` function.

```
filter(ACEDextract,Cond_code!="control")
```

```
# A tibble: 230 x 29
   SubjID Session Cond_code     Sequencing Feedback Correct Incorrect Reamaining
   <chr>  <chr>   <chr>         <chr>      <chr>      <dbl>     <dbl>      <dbl>
 1 S052   c01     adaptive_acc  Adaptive   Accuracy      20        14         29
 2 S053   c01     adaptive_full Adaptive   Full          12        51          0
 3 S054   c01     linear_full   Linear     Full          34        29          0
 4 S055   c01     adaptive_acc  Adaptive   Accuracy       8        55          0
 5 S056   c01     adaptive_full Adaptive   Full          20        43          0
 6 S057   c01     linear_full   Linear     Full          25        28         10
 7 S058   c01     adaptive_acc  Adaptive   Accuracy      33        30          0
 8 S059   c01     adaptive_full Adaptive   Full          20        37          6
 9 S060   c01     linear_full   Linear     Full          28        35          0
10 S061   c01     adaptive_acc  Adaptive   Accuracy      21        42          0
# i 220 more rows
# i 21 more variables: ElapsedTime <dbl>, Gender <chr>, Race <dbl>,
#   Level_Code <chr>, pre_scaled <dbl>, post_scaled <dbl>, Form_Order <dbl>,
#   EAP.sgp <dbl>, EAP.cr <dbl>, EAP.dt <dbl>, EAP.eg <dbl>, EAP.exp <dbl>,
#   EAP.ext <dbl>, EAP.mod <dbl>, EAP.rr <dbl>, EAP.tab <dbl>, EAP.vr <dbl>,
#   EAP.pic <dbl>, P.sgp..H <dbl>, P.sgp..M <dbl>, P.sgp..L <dbl>
```

Chaining data wrangling steps.

The pipe operator `%>%` (from the `magrittr` package) allows chaining the output of one command to another.

It is "syntactic sugar" which tells R to use the output of the last command as the first argument to the next.

In `tidyverse` is used to pass the data set along.

Often used with -> assignment to specify where to save the output.

> **Caution**
>
> The %>% operator needs to be the last thing on the line, not the first on the next line.

```
ACEDextract %>%
  filter(Cond_code != "control") %>%
  select(all_of(c("Cond_code","pre_scaled","post_scaled","EAP.sgp"))) ->
  ACEDworking
summary(ACEDworking)
```

```
 Cond_code           pre_scaled       post_scaled         EAP.sgp
Length:230         Min.   :-999.00   Min.   :-999.00   Min.   :0.0000
Class :character   1st Qu.:  44.00   1st Qu.:  47.00   1st Qu.:0.0020
Mode  :character   Median :  50.00   Median :  54.00   Median :0.1595
                   Mean   :  41.11   Mean   :  46.26   Mean   :0.5464
                   3rd Qu.:  57.00   3rd Qu.:  63.00   3rd Qu.:0.9938
                   Max.   :  78.00   Max.   :  84.00   Max.   :1.9980
```

For more possibilities, look at the `Data Transformation with dplyr` cheat sheet, or *R 4 Data Science* book.

Cleaning the Data

Usually need to do extra work for the raw data.

Missing Data

Look at the summaries. Notice that the minimum value is -999. That is a ridiculous number. In fact, it is the code used for missing values in this data set.

```
summary(ACEDextract)
```

```
   SubjID             Session           Cond_code          Sequencing
Length:290         Length:290         Length:290         Length:290
Class :character   Class :character   Class :character   Class :character
Mode  :character   Mode  :character   Mode  :character   Mode  :character
```

```
  Feedback            Correct          Incorrect          Reamaining
Length:290         Min.   :-999.0   Min.   :-999.00   Min.   :-999.0
Class :character   1st Qu.:   8.0   1st Qu.:  18.00   1st Qu.:   0.0
Mode  :character   Median :  18.0   Median :  34.00   Median :   0.0
                   Mean   :-188.9   Mean   :-177.24   Mean   :-204.1
                   3rd Qu.:  27.0   3rd Qu.:  44.75   3rd Qu.:   0.0
                   Max.   :  51.0   Max.   :  59.00   Max.   :  43.0
 ElapsedTime        Gender              Race           Level_Code
Min.   :-999.0   Length:290         Min.   :-999.000   Length:290
1st Qu.:-999.0   Class :character   1st Qu.:   3.000   Class :character
Median :1506.0   Mode  :character   Median :   6.000   Mode  :character
Mean   : 896.7                      Mean   :  -4.807
3rd Qu.:2093.2                      3rd Qu.:   7.000
Max.   :2693.0                      Max.   :   8.000
  pre_scaled       post_scaled        Form_Order       EAP.sgp
Min.   :-999.00   Min.   :-999.00   Min.   :1.000   Min.   :-999.0000
1st Qu.:  44.00   1st Qu.:  47.00   1st Qu.:1.000   1st Qu.:   0.0000
Median :  50.00   Median :  54.00   Median :1.000   Median :   0.0155
Mean   :  42.72   Mean   :  47.31   Mean   :1.486   Mean   :-206.2563
3rd Qu.:  57.00   3rd Qu.:  61.00   3rd Qu.:2.000   3rd Qu.:   0.7760
Max.   :  78.00   Max.   :  84.00   Max.   :2.000   Max.   :   1.9980
    EAP.cr            EAP.dt            EAP.eg            EAP.exp
Min.   :-999.000   Min.   :-999.0000   Min.   :-999.000   Min.   :-999.000
1st Qu.:   0.439   1st Qu.:   0.4220   1st Qu.:   0.010   1st Qu.:   0.003
Median :   1.008   Median :   0.4270   Median :   0.017   Median :   0.014
Mean   :-205.678   Mean   :-206.2295   Mean   :-206.349   Mean   :-206.612
3rd Qu.:   1.822   3rd Qu.:   0.6698   3rd Qu.:   0.217   3rd Qu.:   0.045
Max.   :   2.000   Max.   :   0.9520   Max.   :   1.935   Max.   :   1.593
   EAP.ext            EAP.mod            EAP.rr
Min.   :-999.0000   Min.   :-999.0000   Min.   :-999.0000
1st Qu.:   0.1628   1st Qu.:   0.0102   1st Qu.:   0.1040
Median :   1.1435   Median :   0.0545   Median :   0.4985
Mean   :-205.6309   Mean   :-206.3877   Mean   :-206.1130
3rd Qu.:   1.9163   3rd Qu.:   0.4305   3rd Qu.:   0.8668
Max.   :   2.0000   Max.   :   1.6970   Max.   :   1.9740
   EAP.tab            EAP.vr            EAP.pic           P.sgp..H
Min.   :-999.0000   Min.   :-999.0000   Min.   :-999.000   Min.   :-999.0000
1st Qu.:   0.0462   1st Qu.:   0.0540   1st Qu.:   0.019   1st Qu.:   0.0000
Median :   0.2780   Median :   0.2060   Median :   0.047   Median :   0.0000
Mean   :-206.1436   Mean   :-206.3237   Mean   :-206.443   Mean   :-206.5617
```

```
 3rd Qu.:   0.9348   3rd Qu.:   0.5235   3rd Qu.:   0.241   3rd Qu.:   0.0318
 Max.   :   1.9740   Max.   :   1.7980   Max.   :   1.895   Max.   :   0.9980
    P.sgp..M            P.sgp..L
 Min.   :-999.0000   Min.   :-999.000
 1st Qu.:   0.0000   1st Qu.:   0.000
 Median :   0.0130   Median :   0.518
 Mean   :-206.5124   Mean   :-206.202
 3rd Qu.:   0.2725   3rd Qu.:   0.993
 Max.   :   0.9030   Max.   :   1.000
```

There are two ways to fix this. First, we can simply set the value of the cell to `NA`. We can do this either by setting the value to `NA` or setting `is.na` to true.

```
ACEDextract$EAP.sgp[ACEDextract$EAP.sgp < -10] <- NA
is.na(ACEDextract$EAP.exp[ACEDextract$EAP.exp < -10]) <- TRUE
```

However, as we need to do every single column, it is probably easier to do this when we read the data in.

```
ACEDextract <- read_csv("ACED_extract1.csv",na="-999")
```

```
Rows: 290 Columns: 29
-- Column specification --------------------------------------------------------
Delimiter: ","
chr  (7): SubjID, Session, Cond_code, Sequencing, Feedback, Gender, Level_Code
dbl (22): Correct, Incorrect, Reamaining, ElapsedTime, Race, pre_scaled, pos...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
summary(ACEDextract)
```

```
   SubjID            Session           Cond_code          Sequencing
 Length:290         Length:290         Length:290         Length:290
 Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character
```

```
  Feedback            Correct        Incorrect       Reamaining
Length:290         Min.   : 2.00   Min.   : 7.00   Min.   : 0.000
Class :character   1st Qu.:14.00   1st Qu.:29.00   1st Qu.: 0.000
Mode  :character   Median :21.00   Median :39.00   Median : 0.000
                   Mean   :22.47   Mean   :37.13   Mean   : 3.274
                   3rd Qu.:30.00   3rd Qu.:46.00   3rd Qu.: 0.000
                   Max.   :51.00   Max.   :59.00   Max.   :43.000
                   NA's   :60      NA's   :60      NA's   :60
 ElapsedTime      Gender              Race        Level_Code
Min.   :  52   Length:290         Min.   :1.000   Length:290
1st Qu.:1542   Class :character   1st Qu.:3.000   Class :character
Median :1939   Mode  :character   Median :6.000   Mode  :character
Mean   :1894                      Mean   :5.585
3rd Qu.:2323                      3rd Qu.:7.000
Max.   :2693                      Max.   :8.000
NA's   :100                       NA's   :3
  pre_scaled     post_scaled      Form_Order       EAP.sgp
Min.   :27.00   Min.   :27.00   Min.   :1.000   Min.   :0.0000
1st Qu.:44.00   1st Qu.:47.00   1st Qu.:1.000   1st Qu.:0.0020
Median :50.00   Median :54.00   Median :1.000   Median :0.1595
Mean   :49.96   Mean   :54.58   Mean   :1.486   Mean   :0.5464
3rd Qu.:57.00   3rd Qu.:61.00   3rd Qu.:2.000   3rd Qu.:0.9938
Max.   :78.00   Max.   :84.00   Max.   :2.000   Max.   :1.9980
NA's   :2       NA's   :2                       NA's   :60
    EAP.cr          EAP.dt          EAP.eg          EAP.exp
Min.   :0.4390   Min.   :0.4220   Min.   :0.0100   Min.   :0.00100
1st Qu.:0.7252   1st Qu.:0.4230   1st Qu.:0.0100   1st Qu.:0.00800
Median :1.3990   Median :0.4735   Median :0.0360   Median :0.02000
Mean   :1.2752   Mean   :0.5802   Mean   :0.4301   Mean   :0.09789
3rd Qu.:1.9378   3rd Qu.:0.7365   3rd Qu.:0.4560   3rd Qu.:0.06925
Max.   :2.0000   Max.   :0.9520   Max.   :1.9350   Max.   :1.59300
NA's   :60       NA's   :60       NA's   :60       NA's   :60
   EAP.ext         EAP.mod         EAP.rr          EAP.tab
Min.   :0.0280   Min.   :0.0040   Min.   :0.1040   Min.   :0.0180
1st Qu.:0.8363   1st Qu.:0.0290   1st Qu.:0.2440   1st Qu.:0.1273
Median :1.4315   Median :0.1140   Median :0.5195   Median :0.5095
Mean   :1.3350   Mean   :0.3807   Mean   :0.7271   Mean   :0.6885
3rd Qu.:1.9658   3rd Qu.:0.6552   3rd Qu.:0.9795   3rd Qu.:1.1473
Max.   :2.0000   Max.   :1.6970   Max.   :1.9740   Max.   :1.9740
NA's   :60       NA's   :60       NA's   :60       NA's   :60
    EAP.vr          EAP.pic         P.sgp..H         P.sgp..M
Min.   :0.0290   Min.   :0.0110   Min.   :0.00000   Min.   :0.0000
```

```
1st Qu.:0.1492   1st Qu.:0.0350   1st Qu.:0.00000   1st Qu.:0.0020
Median :0.2750   Median :0.0830   Median :0.00000   Median :0.0575
Mean   :0.4615   Mean   :0.3114   Mean   :0.16140   Mean   :0.2235
3rd Qu.:0.6368   3rd Qu.:0.3415   3rd Qu.:0.07475   3rd Qu.:0.3975
Max.   :1.7980   Max.   :1.8950   Max.   :0.99800   Max.   :0.9030
NA's   :60       NA's   :60       NA's   :60        NA's   :60
   P.sgp..L
Min.   :0.0000
1st Qu.:0.0810
Median :0.8415
Mean   :0.6150
3rd Qu.:0.9980
Max.   :1.0000
NA's   :60
```

Coding categorical variables

The `read_csv` function doesn't do anything in particular with strings.

For `SubjID` that is sensible, that variable is only really a label.

However, `Session`, `Cond_code`, `Feedback`, `Gender`, `Race` and `Level_Code` are actually categorical (nominal) variables. In R, these are called factor variables, and the function `factor` is used to create them.

```
ACEDextract$Session <- factor(ACEDextract$Session)
ACEDextract$Cond_code <- factor(ACEDextract$Cond_code)
ACEDextract$Sequencing <- factor(ACEDextract$Sequencing)
ACEDextract$Feedback <- factor(ACEDextract$Feedback)
ACEDextract$Gender <- factor(ACEDextract$Gender)
ACEDextract$Race <- factor(ACEDextract$Race,1:8)
ACEDextract$Level_Code <- factor(ACEDextract$Level_Code)
summary(ACEDextract)
```

```
   SubjID             Session           Cond_code     Sequencing
Length:290         c02    : 18   adaptive_acc :81   Adaptive:158
Class :character   c03    : 18   adaptive_full:77   Linear  : 72
Mode  :character   c04    : 18   control      :60   NA's    : 60
                   c13    : 18   linear_full  :72
                   c01    : 17
                   (Other):141
                   NA's   : 60
```

```
    Feedback      Correct        Incorrect       Reamaining      ElapsedTime
Accuracy: 81   Min.   : 2.00   Min.   : 7.00   Min.   : 0.000   Min.   :  52
Full    :149   1st Qu.:14.00   1st Qu.:29.00   1st Qu.: 0.000   1st Qu.:1542
NA's    : 60   Median :21.00   Median :39.00   Median : 0.000   Median :1939
               Mean   :22.47   Mean   :37.13   Mean   : 3.274   Mean   :1894
               3rd Qu.:30.00   3rd Qu.:46.00   3rd Qu.: 0.000   3rd Qu.:2323
               Max.   :51.00   Max.   :59.00   Max.   :43.000   Max.   :2693
               NA's   :60      NA's   :60      NA's   :60       NA's   :100
   Gender         Race        Level_Code    pre_scaled     post_scaled
Female:144   7      :113   Academic:165   Min.   :27.00   Min.   :27.00
Male  :146   6      : 65   ELL     : 22   1st Qu.:44.00   1st Qu.:47.00
             3      : 43   Honors  : 38   Median :50.00   Median :54.00
             2      : 27   Part 1  : 30   Mean   :49.96   Mean   :54.58
             8      : 21   Part 2  :  8   3rd Qu.:57.00   3rd Qu.:61.00
             (Other): 18   Regular : 27   Max.   :78.00   Max.   :84.00
             NA's   :  3                  NA's   :2       NA's   :2
  Form_Order       EAP.sgp          EAP.cr           EAP.dt
Min.   :1.000   Min.   :0.0000   Min.   :0.4390   Min.   :0.4220
1st Qu.:1.000   1st Qu.:0.0020   1st Qu.:0.7252   1st Qu.:0.4230
Median :1.000   Median :0.1595   Median :1.3990   Median :0.4735
Mean   :1.486   Mean   :0.5464   Mean   :1.2752   Mean   :0.5802
3rd Qu.:2.000   3rd Qu.:0.9938   3rd Qu.:1.9378   3rd Qu.:0.7365
Max.   :2.000   Max.   :1.9980   Max.   :2.0000   Max.   :0.9520
                NA's   :60       NA's   :60       NA's   :60
    EAP.eg          EAP.exp          EAP.ext          EAP.mod
Min.   :0.0100   Min.   :0.00100   Min.   :0.0280   Min.   :0.0040
1st Qu.:0.0100   1st Qu.:0.00800   1st Qu.:0.8363   1st Qu.:0.0290
Median :0.0360   Median :0.02000   Median :1.4315   Median :0.1140
Mean   :0.4301   Mean   :0.09789   Mean   :1.3350   Mean   :0.3807
3rd Qu.:0.4560   3rd Qu.:0.06925   3rd Qu.:1.9658   3rd Qu.:0.6552
Max.   :1.9350   Max.   :1.59300   Max.   :2.0000   Max.   :1.6970
NA's   :60       NA's   :60        NA's   :60       NA's   :60
    EAP.rr          EAP.tab          EAP.vr          EAP.pic
Min.   :0.1040   Min.   :0.0180   Min.   :0.0290   Min.   :0.0110
1st Qu.:0.2440   1st Qu.:0.1273   1st Qu.:0.1492   1st Qu.:0.0350
Median :0.5195   Median :0.5095   Median :0.2750   Median :0.0830
Mean   :0.7271   Mean   :0.6885   Mean   :0.4615   Mean   :0.3114
3rd Qu.:0.9795   3rd Qu.:1.1473   3rd Qu.:0.6368   3rd Qu.:0.3415
Max.   :1.9740   Max.   :1.9740   Max.   :1.7980   Max.   :1.8950
NA's   :60       NA's   :60       NA's   :60       NA's   :60
   P.sgp..H         P.sgp..M         P.sgp..L
Min.   :0.00000   Min.   :0.0000   Min.   :0.0000
1st Qu.:0.00000   1st Qu.:0.0020   1st Qu.:0.0810
```

```
 Median :0.00000   Median :0.0575   Median :0.8415
 Mean   :0.16140   Mean   :0.2235   Mean   :0.6150
 3rd Qu.:0.07475   3rd Qu.:0.3975   3rd Qu.:0.9980
 Max.   :0.99800   Max.   :0.9030   Max.   :1.0000
 NA's   :60        NA's   :60       NA's   :60
```

Making a gain score

We can add new variables to the tibble using `mutate()`. Don't forget to assign the result back to a variable.

```
ACEDextract %>%
  mutate(gain=post_scaled-pre_scaled) ->
  ACEDextract
summary(ACEDextract$gain)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
-22.000  -1.250   5.000   4.618  10.000  34.000       2
```

```
haven::write_sav(ACEDextract,"ACEDextract.sav")
```

Marginal Summaries

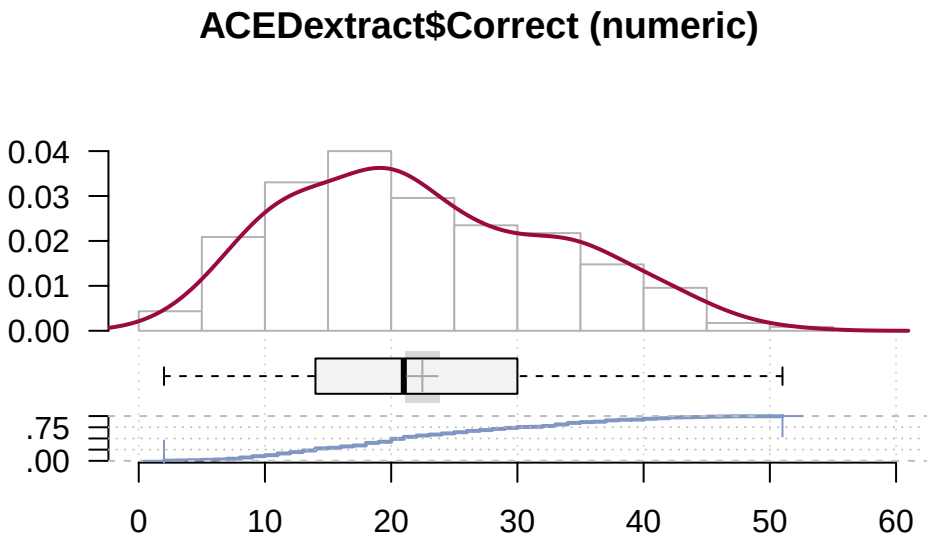
Descriptive Summaries

The function `DescTools::Desc` gives a bunch of summaries.

```
Desc(ACEDextract$Correct)
```

```
------------------------------------------------------------------------------
ACEDextract$Correct (numeric)

  length      n    NAs  unique     0s   mean  meanCI'
     290    230     60      47      0  22.47   21.10
          79.3%  20.7%          0.0%          23.84

     .05    .10    .25  median    .75    .90     .95
```

```
    8.00    9.00   14.00    21.00   30.00   37.00    41.00

   range      sd   vcoef      mad     IQR    skew     kurt
   49.00   10.55    0.47    11.86   16.00    0.38    -0.68

lowest : 2.0, 3.0, 4.0, 5.0 (2), 6.0 (2)
highest: 44.0, 45.0 (2), 46.0, 47.0, 51.0

' 95%-CI (classic)
```




```
ACEDextract %>%
  summarize(mCorrect=mean(Correct), sdCorrect=sd(Correct),
            mIncorrect=mean(Incorrect),sdIncorrect=sd(Incorrect))
```

```
# A tibble: 1 x 4
  mCorrect sdCorrect mIncorrect sdIncorrect
     <dbl>     <dbl>      <dbl>       <dbl>
1       NA        NA         NA          NA
```

Need to filter out NAs

```
ACEDextract %>%
  filter(!is.na(Correct) & !is.na(Incorrect)) %>%
  summarize(mCorrect=mean(Correct), sdCorrect=sd(Correct),
```

```
            mIncorrect=mean(Incorrect),sdIncorrect=sd(Incorrect))
```

```
# A tibble: 1 x 4
  mCorrect sdCorrect mIncorrect sdIncorrect
     <dbl>     <dbl>      <dbl>       <dbl>
1     22.5      10.5       37.1        11.7
```

Adding a `group_by` step breaks the summaries down.

```
ACEDextract %>%
  filter(!is.na(Correct) & !is.na(Incorrect)) %>%
  group_by(Level_Code) %>%
  summarize(mCorrect=mean(Correct), sdCorrect=sd(Correct),
            mIncorrect=mean(Incorrect),sdIncorrect=sd(Incorrect))
```

```
# A tibble: 6 x 5
  Level_Code mCorrect sdCorrect mIncorrect sdIncorrect
  <fct>         <dbl>     <dbl>      <dbl>       <dbl>
1 Academic      24         9.30       37.1       10.4
2 ELL            8.53      3.76       37.9       15.1
3 Honors        32.4       9.52       28.0        9.95
4 Part 1        16.7       6.94       46.2        6.97
5 Part 2        11.1       3.58       51.6        3.74
6 Regular       17.5       6.59       37         12.9
```

Plots using ggplot2

GGplot (stands for the *Grammar of Graphics*, a book by Leyland Wilkenson)

It has the following steps:

- A call to `ggplot()` which specifies the data (often piped in using `%>%`)
- A call to `aes()` which sets up the aesthetics of the plot.
  - `x` – the x-axis variable (can be a column name from the data)
  - `y` – the y-axis variable
  - `color` and `fill` – used to add color to the plot
  - `shape` — shape of the points
  - `linetype` – type of the line
  - `size` – size of points/linewidth

  - Could be others specific to certain geometries.
- A call to `geom_XXX()` to show what should be rendered.

Other optional steps: * A call to `stat_XXX()` to calculate statistics * A call to `scale_*_XXX()` to establish a scale, here `*` is an aesthetic. * A call to `facet_grid()` or `facet_wrap()` to put multiple subplots * Calls to `labs()`, `annotate()`, `guide()`, `theme()`.

These are chained together with `+`.

Graph is drawn when the result is printed.

See the `Data Visualization with ggplot2` cheat sheet.

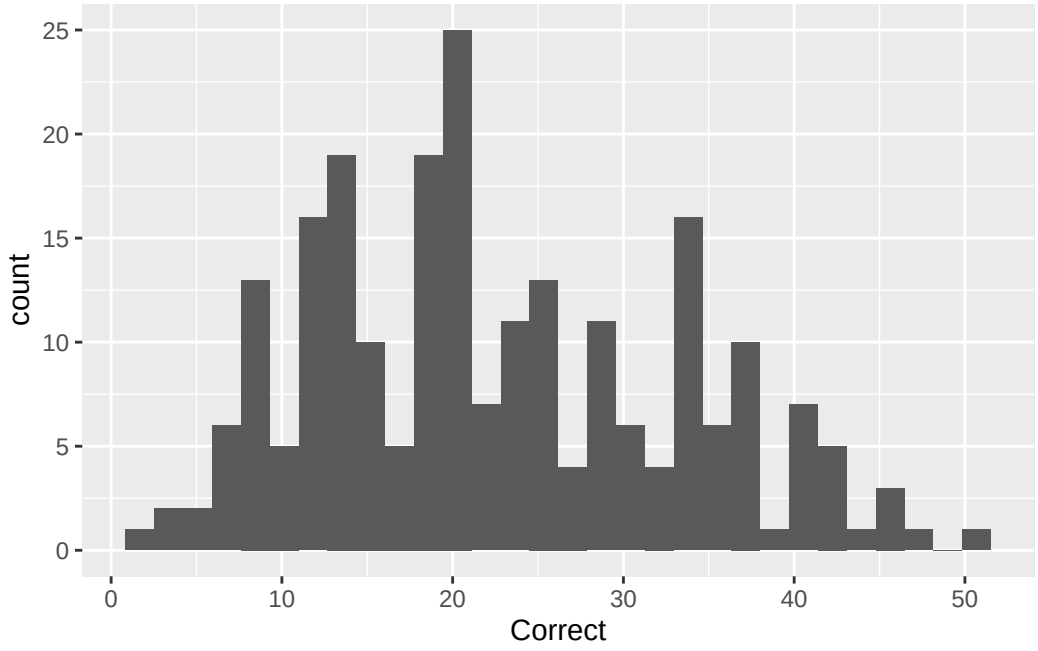
Histograms and Density Plots

Histogram uses a single X value.

```
corhist <- ggplot(ACEDextract,aes(x=Correct)) + geom_histogram()
corhist
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
Warning: Removed 60 rows containing non-finite values (`stat_bin()`).
```



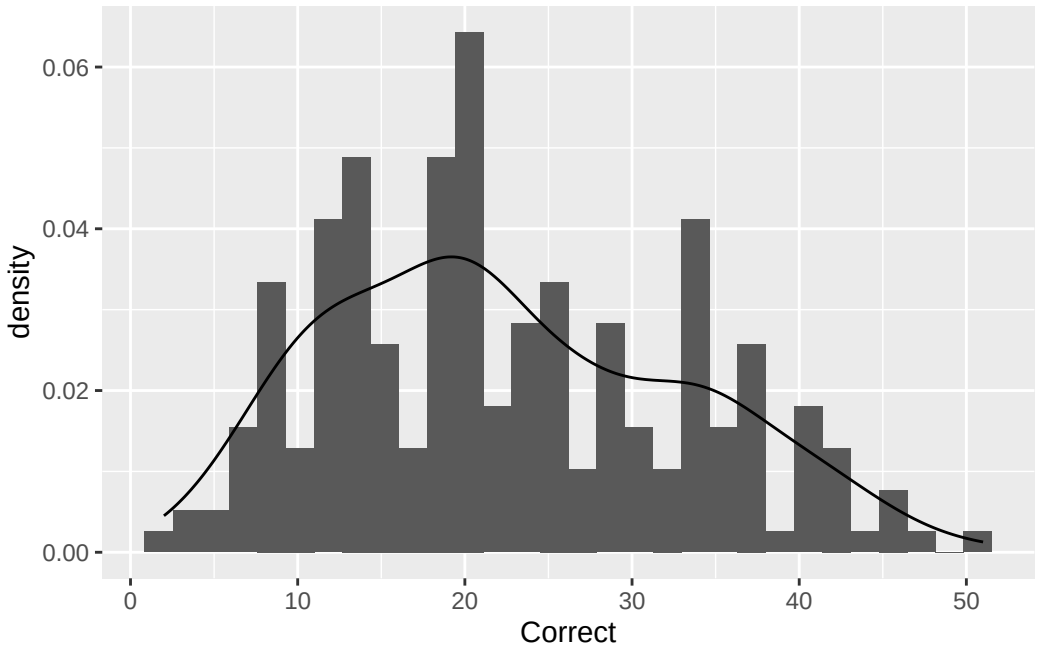
```
ggplot(ACEDextract,aes(x=Correct)) + geom_histogram(aes(y=..density..)) + geom_density()
```

```
Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
i Please use `after_stat(density)` instead.
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
Warning: Removed 60 rows containing non-finite values (`stat_bin()`).
```

```
Warning: Removed 60 rows containing non-finite values (`stat_density()`).
```



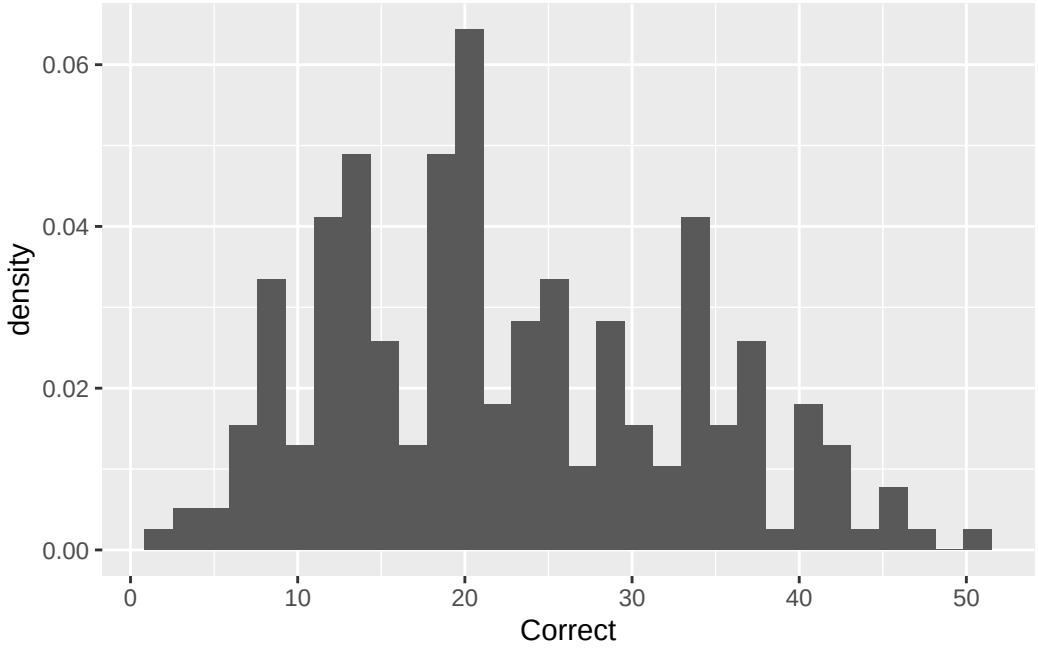
Add a normal curve:

```
ggplot(ACEDextract,aes(x=Correct)) + geom_histogram(aes(y=..density..)) +
  stat_function(fun=dnorm,args=list(mean=mean(ACEDextract$Correct),
                                    sd=sd(ACEDextract$Correct)))
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
Warning: Removed 60 rows containing non-finite values (`stat_bin()`).
```

```
Warning: Removed 101 rows containing missing values (`geom_function()`).
```



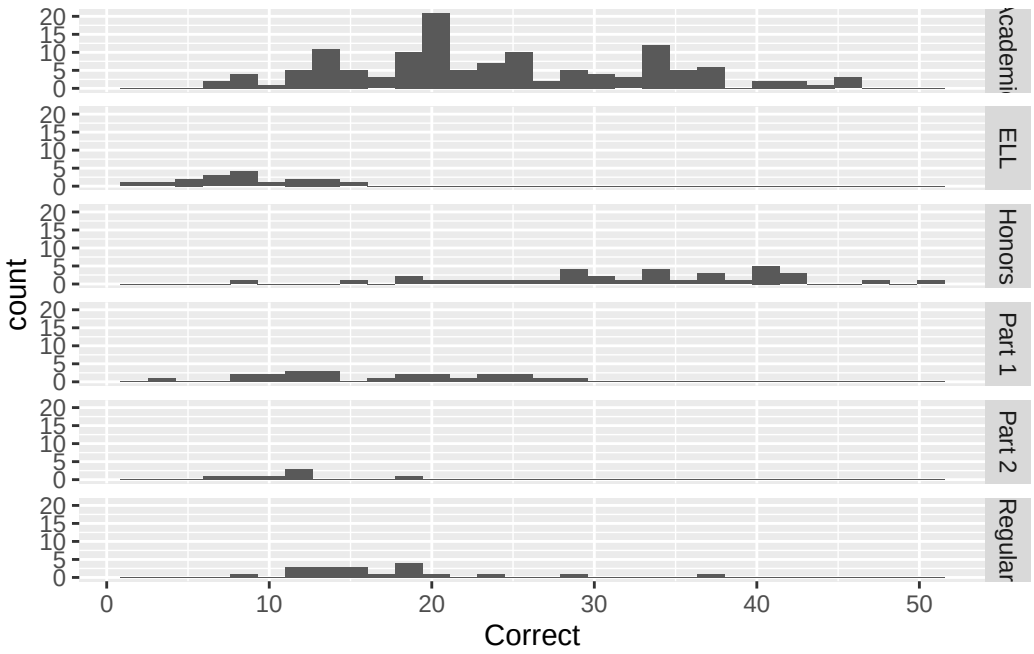
Using Facets to break down by group.

Can use `rows` or `cols` argument to grid to facet by rows or columns.

```
ggplot(ACEDextract,aes(x=Correct)) + geom_histogram() +
  facet_grid(rows=vars(Level_Code))
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
Warning: Removed 60 rows containing non-finite values (`stat_bin()`).
```

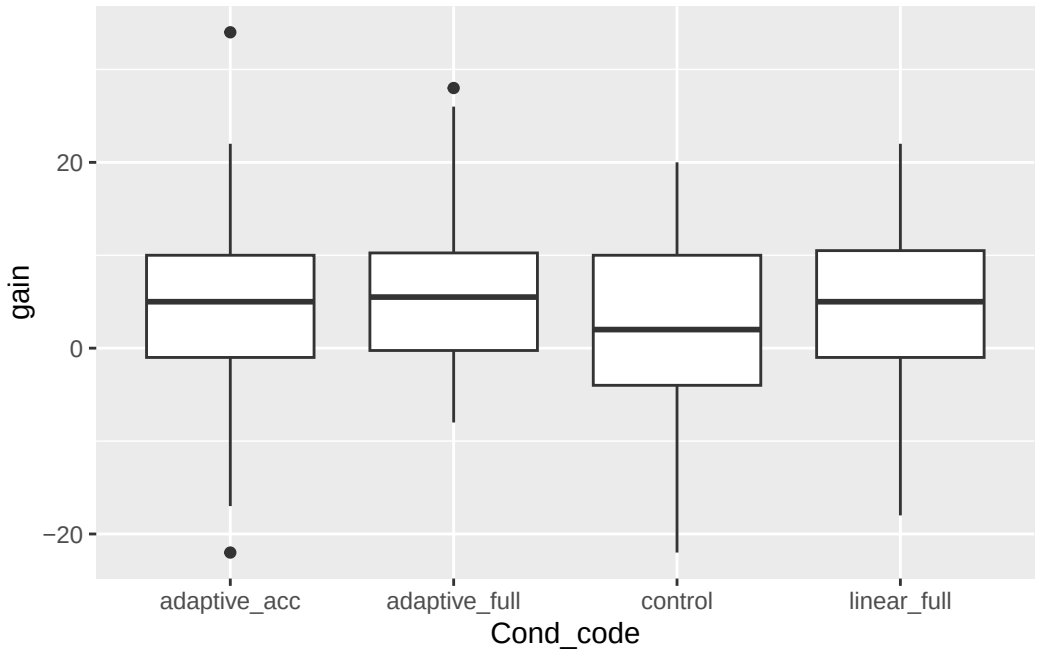


Boxplots, Dot Plots and Violin Plots

Boxplots naturally use a categorical variable on one access and a continuous variable on the other access.

```
ggplot(ACEDextract,aes(x=Cond_code,y=gain)) + geom_boxplot()
```

```
Warning: Removed 2 rows containing non-finite values (`stat_boxplot()`).
```

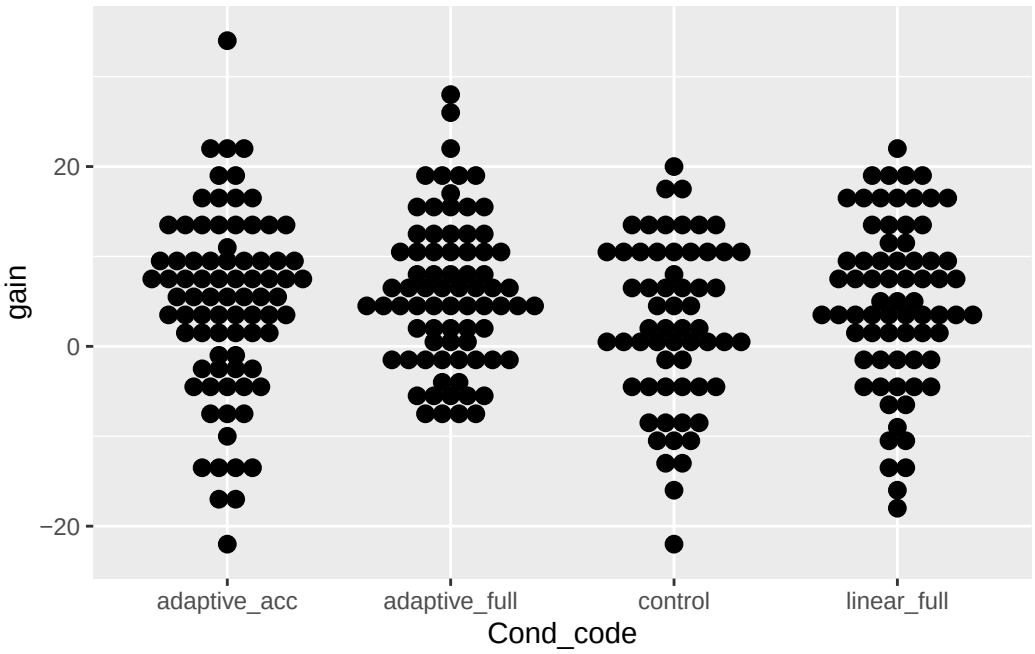


Dotplots plot the individual points, and so give more detail than a boxplot.

```
ggplot(ACEDextract,aes(x=Cond_code,y=gain))+
  geom_dotplot(binaxis="y",stackdir="center")
```

```
Bin width defaults to 1/30 of the range of the data. Pick better value with
`binwidth`.
```

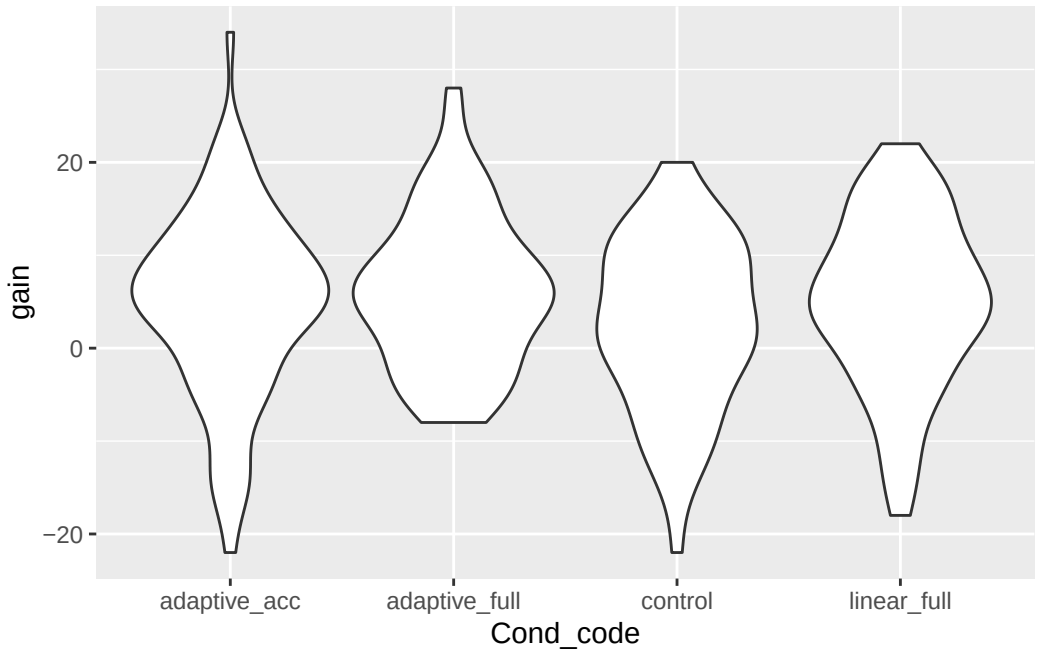
```
Warning: Removed 2 rows containing missing values (`stat_bindot()`).
```



A violin plot takes the density and doubles it to produce a shape like an odd stringed instrument.

```
ggplot(ACEDextract,aes(x=Cond_code,y=gain)) + geom_violin()
```

```
Warning: Removed 2 rows containing non-finite values (`stat_ydensity()`).
```



Quantile–Quantile plots

Goal, check if data follow a particular distribution.

1) Sort data in increasing order -

- These are the sample quantiles, corresponding to probabilities $(i - .5)/N$

2) Look up the corresponding quantiles of the reference distribution (e.g., `qnorm()`)
3) Plot them, should be a diagonal line.

- Intercept is mean and slope is sd.

4) Departures from straight line indicate distribution doesn't fit.

This is not a scatterplot.

SPSS

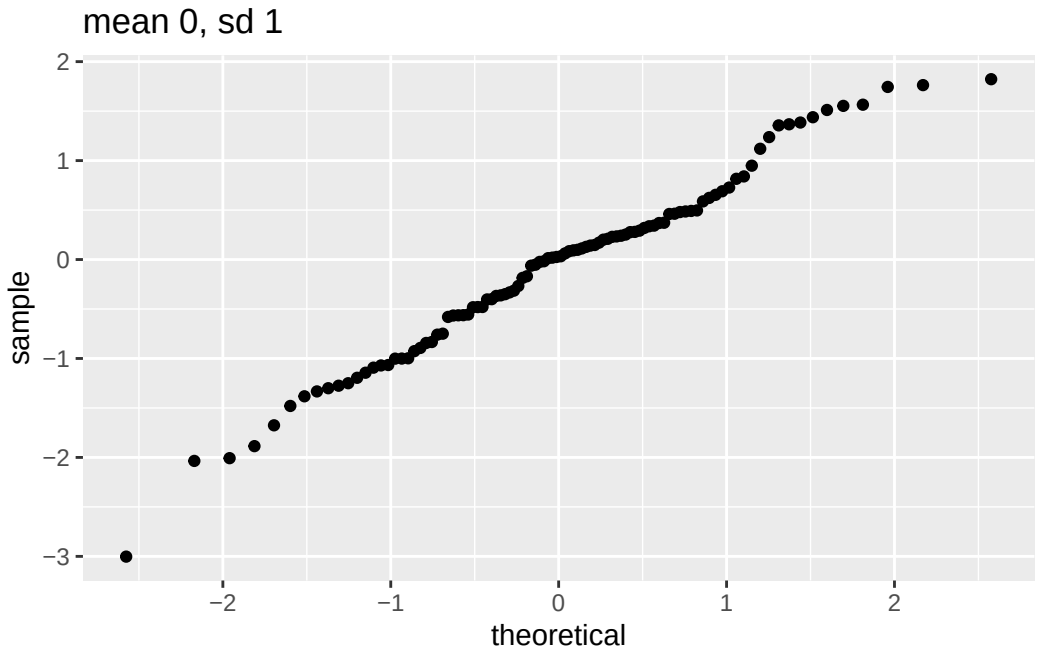
- `Analyze > Descriptive Statistics > Q-Q Plots...`
- Regular and Detrended version
  - I prefer regular

- http://statistics.laerd.com/spss-tutorials/testing-for-normality-using-spss-statistics.php
- http://www.microbiologybytes.com/maths/spss2.html

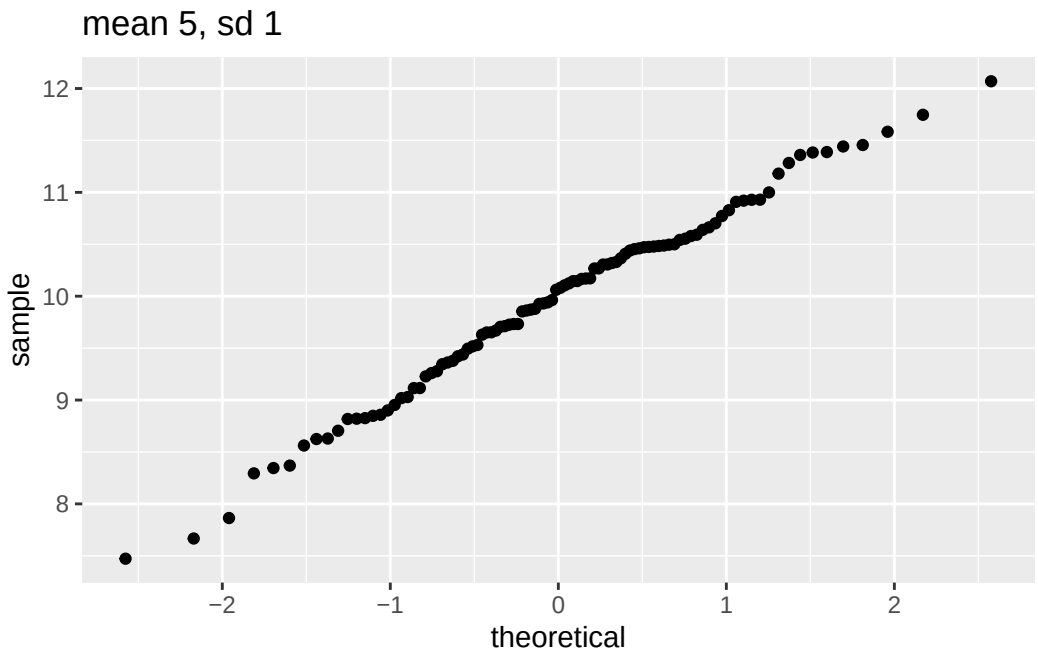
SPSS puts the theoretical quantiles on the Y axis, and R puts them on the X, so the interpretations are flipped.

Mean and standard deviation.

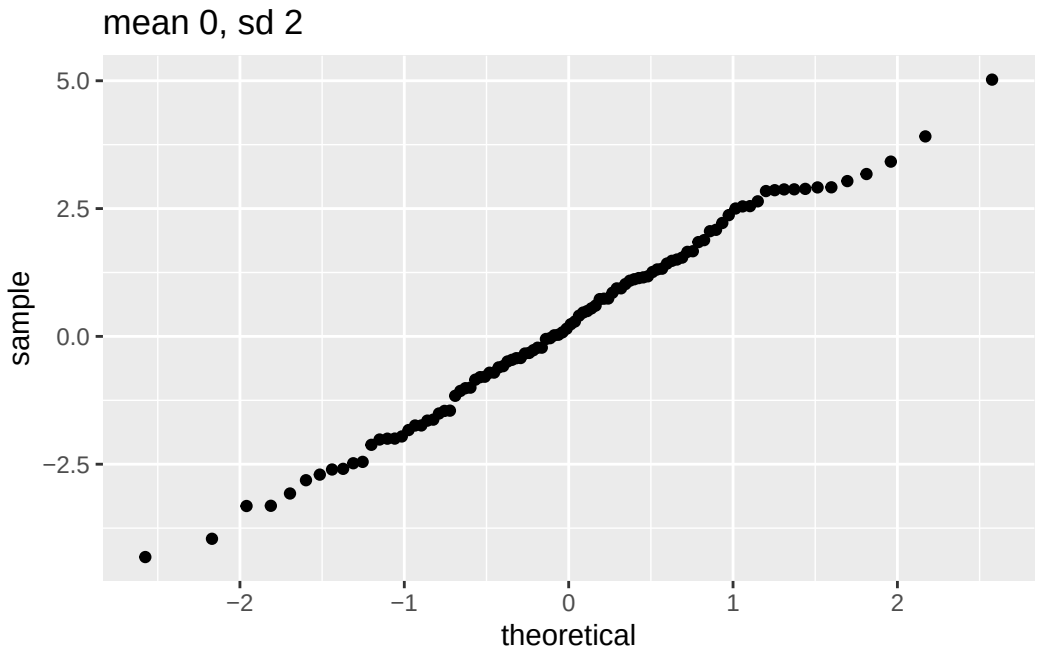
```
normdat <- data.frame(
  norm1 = rnorm(100),
  norm2 = rnorm(100,10),
  norm3 = rnorm(100,0,2),
  norm4 = rnorm(100,5,2.5)
)
ggplot(normdat,aes(sample=norm1)) + geom_qq() + labs(title="mean 0, sd 1")
```



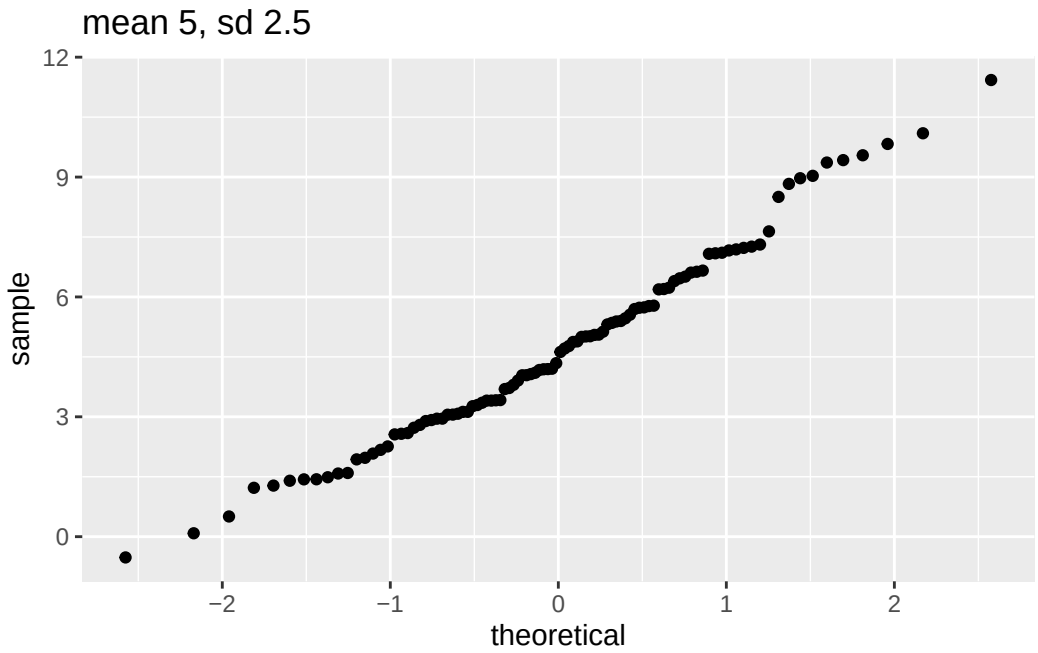

```
ggplot(normdat,aes(sample=norm2)) + geom_qq() + labs(title="mean 5, sd 1")
```




```
ggplot(normdat,aes(sample=norm3)) + geom_qq() + labs(title="mean 0, sd 2")
```

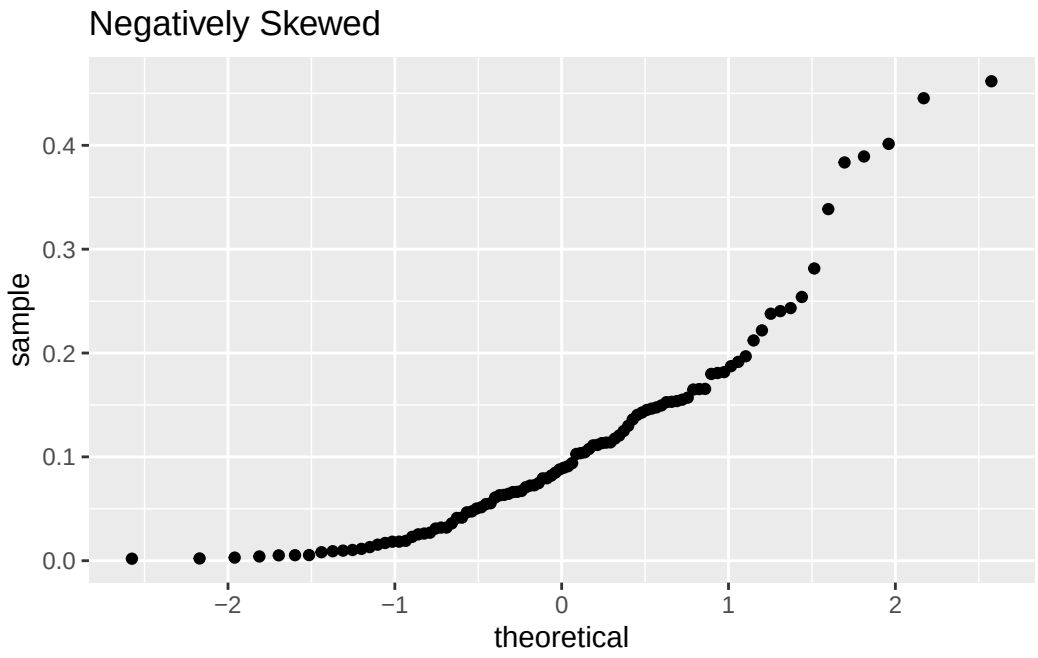


```
ggplot(normdat,aes(sample=norm4)) + geom_qq() + labs(title="mean 5, sd 2.5")
```

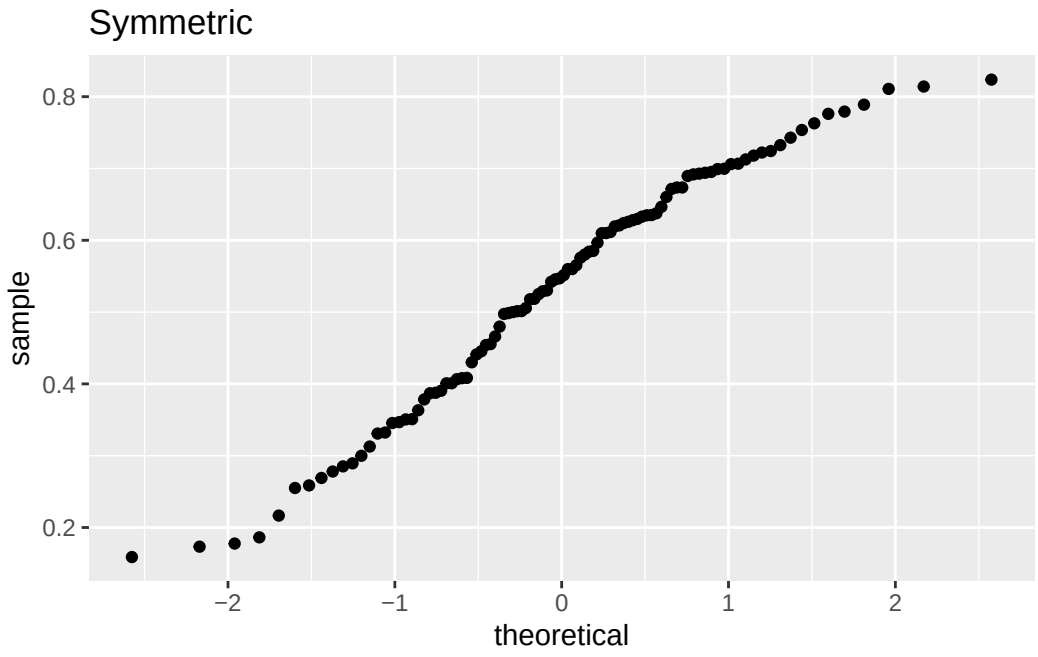


Skewness is a C curve

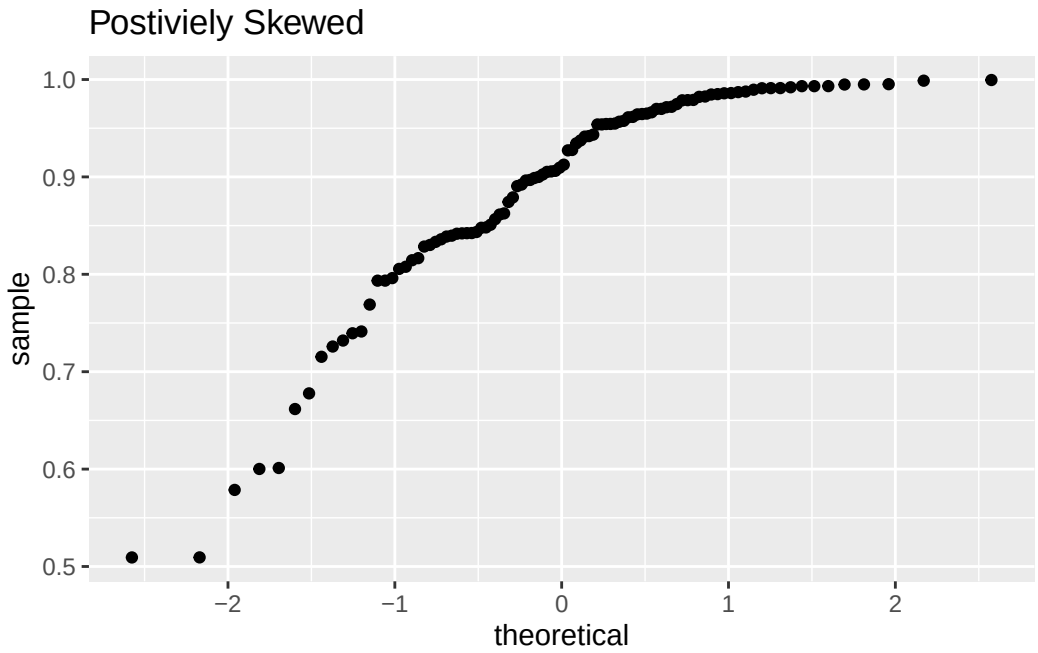
```
skewdat <- data.frame(
  negskew = rbeta(100,1,9),
  symmetric = rbeta(100,5,5),
  posskew = rbeta(100,9,1)
)
ggplot(skewdat,aes(sample=negskew)) + geom_qq() + labs(title="Negatively Skewed")
```




```
ggplot(skewdat,aes(sample=symmetric)) + geom_qq() + labs(title="Symmetric ")
```



```
ggplot(skewdat,aes(sample=posskew)) + geom_qq() + labs(title="Postiviely Skewed")
```

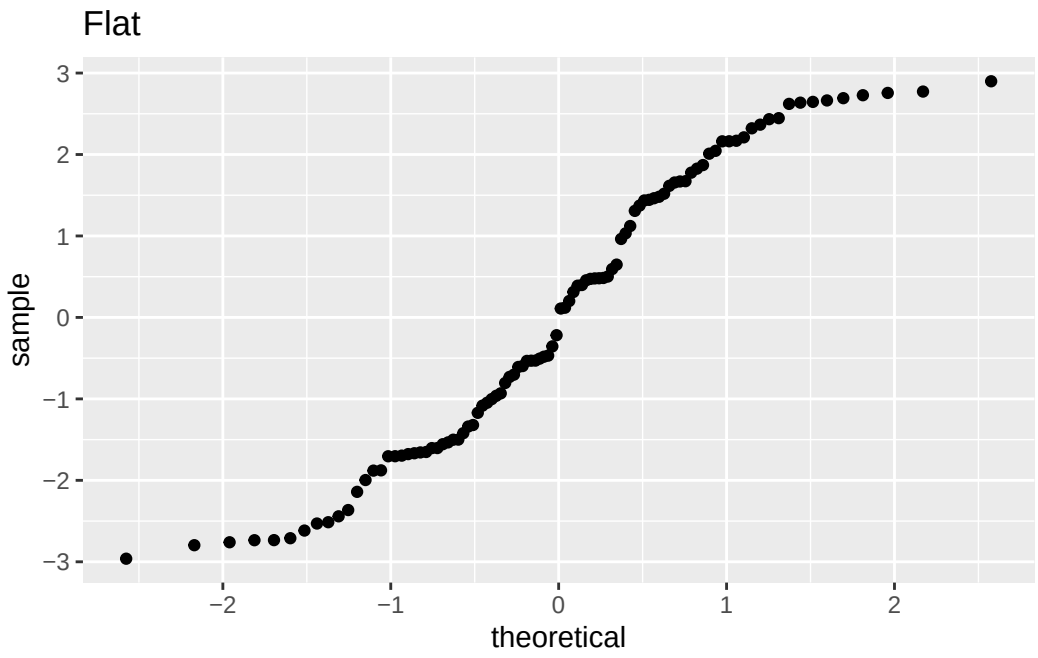


https://pluto.coe.fsu.edu/rdemos/IntroStats/SkewnessQQ.Rmd

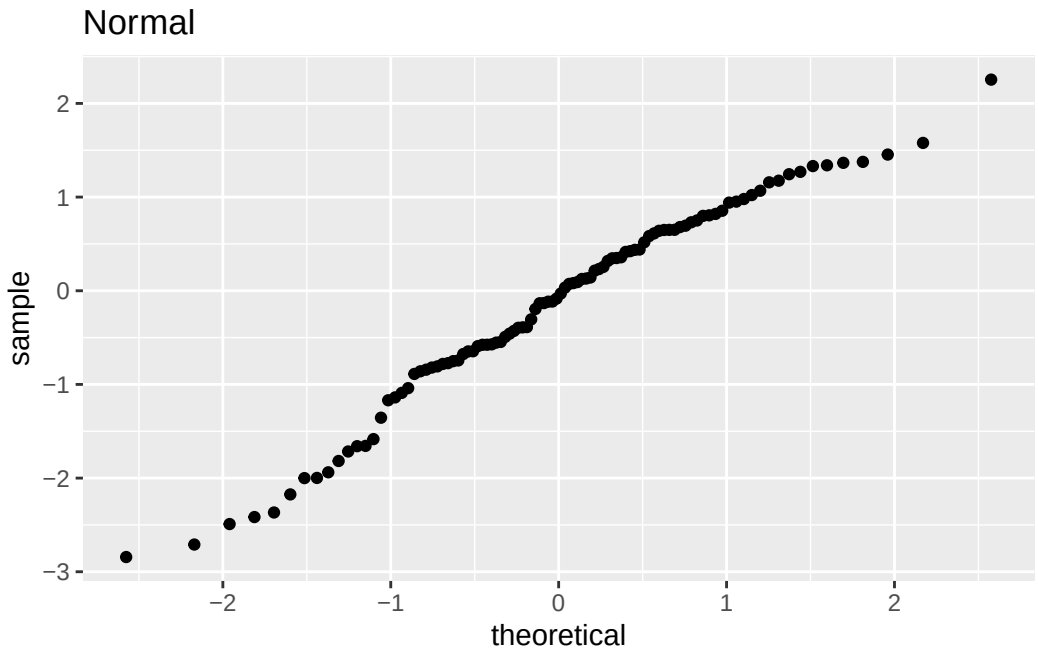
Note: SPSS reverses the axis, so the curves go in the opposite direction

Kurtosis shows up as an S-curve

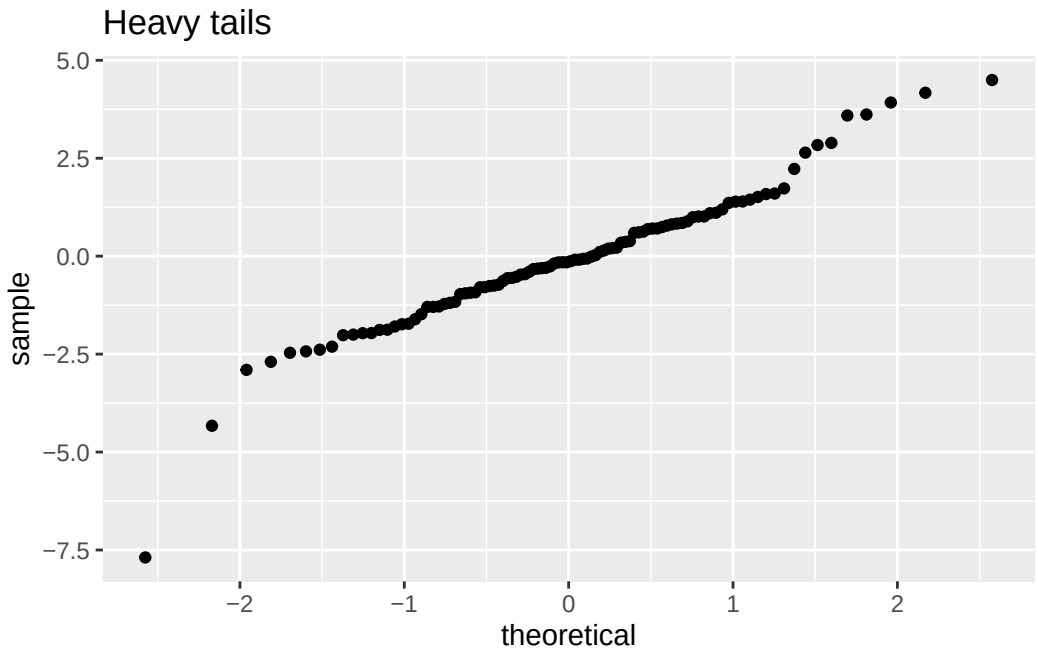
```
kurtdat <- data.frame(
  flat = runif(100,-3,3),
  meso = rnorm(100),
  lepto = rt(100,3)
)
ggplot(kurtdat,aes(sample=flat)) + geom_qq() + labs(title="Flat")
```




```
ggplot(kurtdat,aes(sample=meso)) + geom_qq() + labs(title="Normal")
```



```
ggplot(kurtdat,aes(sample=lepto)) + geom_qq() + labs(title="Heavy tails")
```



Generally, flat is not a problem. Heavy tails means lots of outliers, so estimates could be sensitive to outliers.

https://pluto.coe.fsu.edu/rdemos/IntroStats/KurtosisQQ.Rmd

A short aside on hypothesis testing.

If we estimate a statistic, *from a representative sample*, then 95% of the the population value should be within 2 standard errors of the sample value.

If we want to test that a statistic is not zero, we can divide the test statistic by its standard error. Call this value t. If $t > 2$ then we are getting good evidence that the sample didn't come from a population in which the true value is zero.

Example

```
ACEDextract %>%
  filter(!is.na(gain)) %>%
  group_by(Cond_code) %>%
  summarize(M=mean(gain),S=sd(gain),se=MeanSE(gain),
            t=mean(gain)/MeanSE(gain), d=mean(gain)/sd(gain))
```

```
# A tibble: 4 x 6
  Cond_code          M     S    se     t     d
  <fct>          <dbl> <dbl> <dbl> <dbl> <dbl>
1 adaptive_acc    4.72 10.0  1.11   4.24 0.471
2 adaptive_full   6.04  8.18 0.938  6.44 0.739
3 control         2.45  9.17 1.18   2.07 0.267
4 linear_full     4.82  9.15 1.09   4.44 0.527
```

The t statistic follows a Student's t distribution, with degrees of freedom related to how many data points are available to estimate the S.E.

We want the 97.5% point of the Student's t distribution. The `qt` function calculates this for us.

```
qt(.976,c(1:30,Inf))
```

```
 [1] 13.237770  4.398367  3.235904  2.816470  2.604287  2.477001  2.392389
 [8]  2.332154  2.287123  2.252201  2.224334  2.201586  2.182667  2.166687
[15]  2.153012  2.141176  2.130833  2.121718  2.113623  2.106388  2.099881
[22]  2.093999  2.088655  2.083780  2.079313  2.075206  2.071417  2.067910
[29]  2.064656  2.061627  1.977368
```

So as long as you have 20 or so observations per group, 2 is a pretty good approximation.

Don't want to take significance testing, p-value too seriously, as there are lots of assumptions which may not hold.

Effect size

Note that significance is really a measure of how good the sample is.

If the sample is really big, we can pick up really small differences.

So compute effect sizes.

For gain score, the effect size is Cohen's d, which we get by dividing the mean by the standard deviation.

Quick interpretation table

Test Statistic	Effect Size	Interpretation
Big	Big	Effect is likely big and important
Big	Small	Sample size is big enough to detect difference but it is small and unimportant
Small	Small	Effect is small and unimportant
Small	Big	Might be an effect, but don't have enough power to be sure

Assignment

Pull in the ACED data and do some exploratory analysis.

1) What can you learn about the variables?
2) Are the variables normally distributed?
3) Are there any unusual values that should be investigated?