USC Department Of Translational Genomics
Keck Medicine of USC

TRGN 527: Applied Data Science and Bioinformatics

UNIT II. Descriptive Statistics

Week 5 - Lecture 1

Enrique I. Velazquez Villarreal, M.D., Ph.D., M.P.H., M.S. | **Assistant Professor**

Dept. of Translational Genomics

USC | Keck School of Medicine | Norris Comprehensive Cancer Center

Leader of the USC Bioinformatics Core – *USC CaRE2 Health Equity Center*

David W. Craig, Ph.D. | **Professor and Vice Chair**

Dept. of Translational Genomics

USC | Keck School of Medicine | Norris Comprehensive Cancer Center

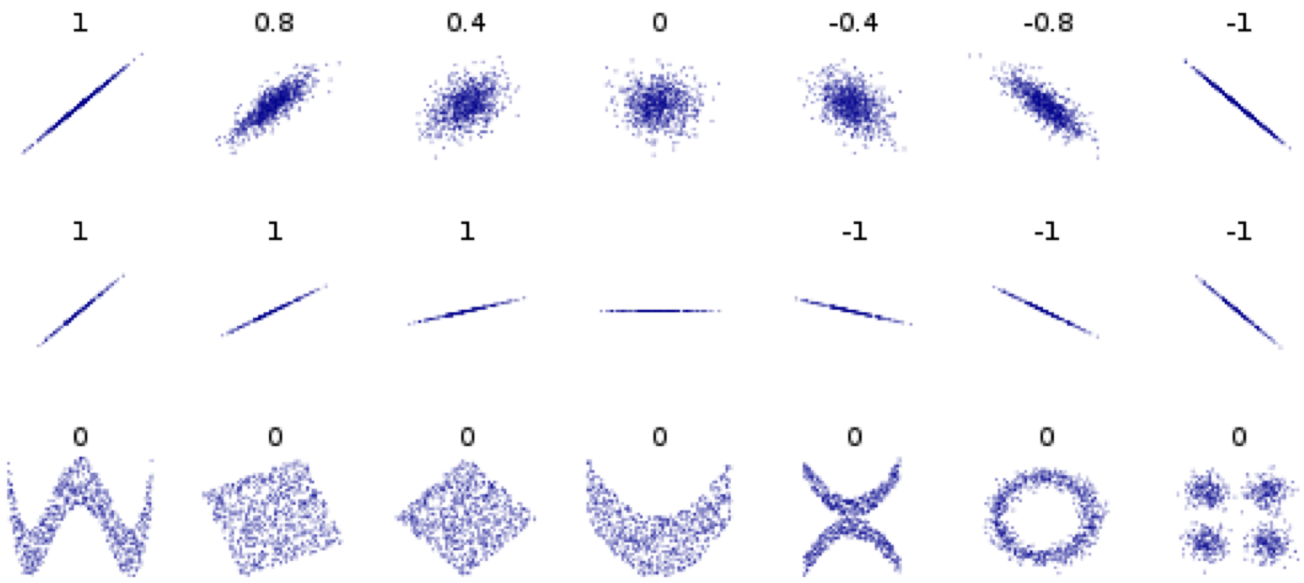
Co-Director, Institute of Translational Genomics

Topics

- Correlations

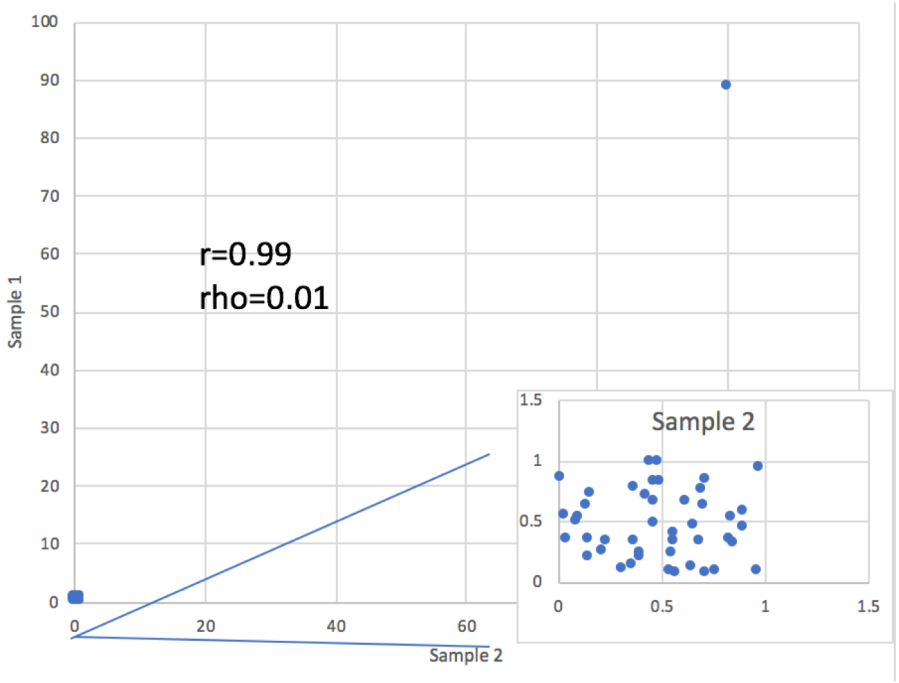
Correlations

- Correlation is any statistical association, though in common usage it most often refers to how close two variables are to having a linear relationship with each other.



Correlations

- Correlation is often described as correlation coefficient – which can be calculated in different ways depending on whether one believes the underlying data is a normally distributed or not.
- When normally distributed, we calculate the Pearson correlation efficient.
- When this is not the case, one can calculate the Spearman which avoids the tail wagging the dog.

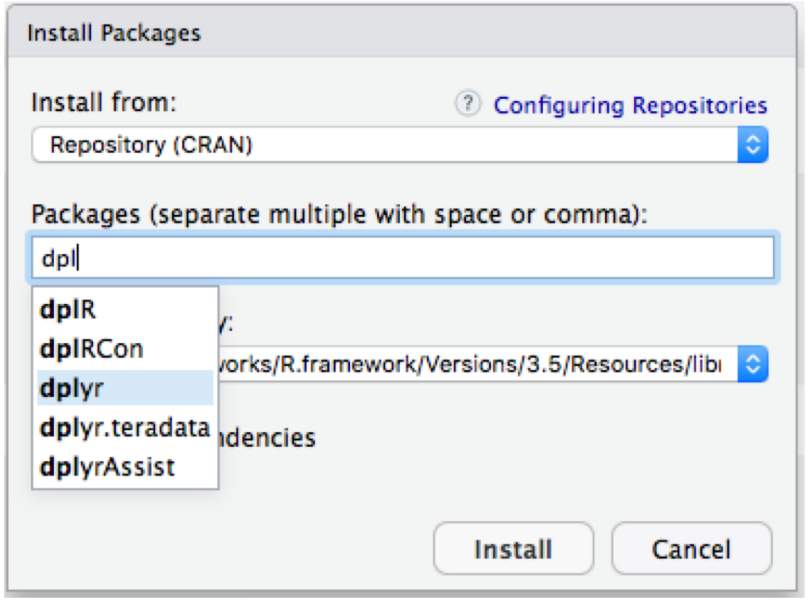


Correlation analysis

- We have our two datasets:
  - expression_results.csv
  - sample_info.csv

Correlations analysis

- Now lets start by loading in our libraries (if dplyr's package isn't installed you should type install.package("dplyr") in the command line window.
- Lets first create an "R" codeblock by clicking "Insert" and then selecting "R" from the pull down.
- Lets load in the files into two dataframes: samples & genes.
- The following should be in your code block.



TRGN599_Week_5_Lecture_1

Enrique I. Velazquez Villarreal, MD, PhD, MPH, MS

2/1/2019

Correlations analysis

```
# Check your current working directory
getwd()
```

```
## [1] "/Users/enriquevelazquez/Documents/R_working_directory"
```

```
# Set your working directory
setwd("/Users/enriquevelazquez/Documents/R_working_directory")
```

Correlations analysis

Uploading datasets

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

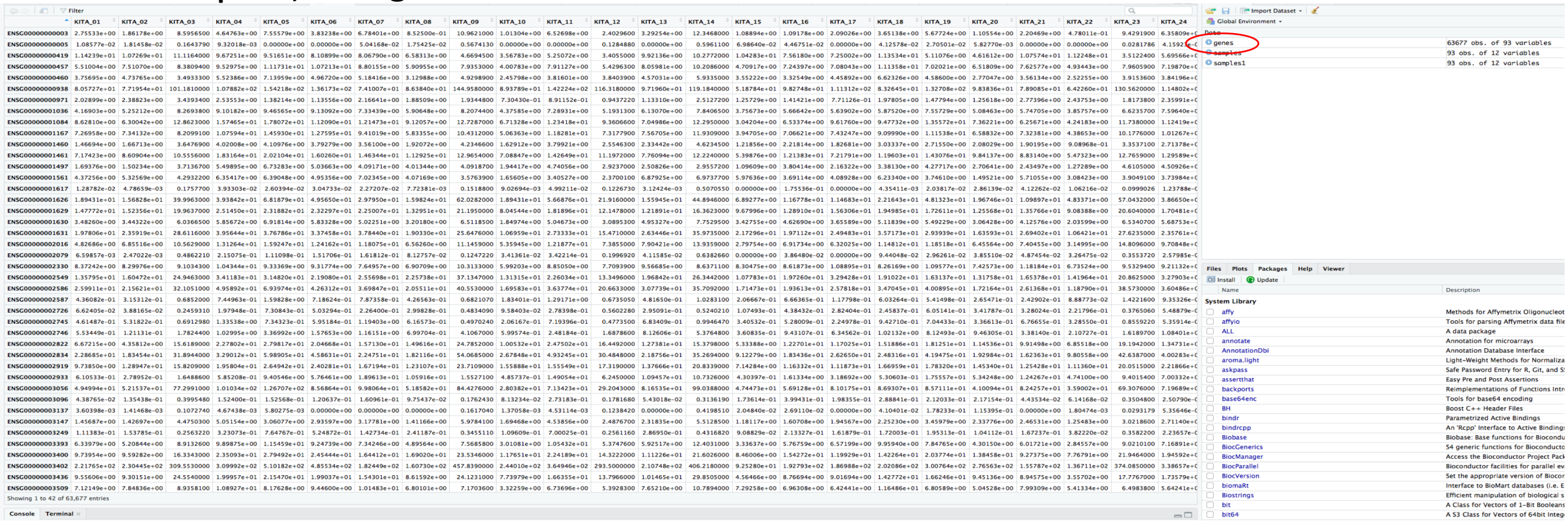
```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
samples <- read.csv('sample_info.csv', header = TRUE, sep = ",", quote = "'", dec = ".", fill = TRUE, row.names =
1)

genes <- read.csv('expression_results.csv',header = TRUE, sep = ",", quote = "'", dec = ".", fill = TRUE, row.nam
es = 1)
```

Correlations

- Press the green arrow in the upper right side to run the code block. There may be a few notes involving dplyr. You should see in the upper right there are two dataframes for samples and genes.
- We can click on each to get a preview.
- For examples, the genes dataframe is shown below.

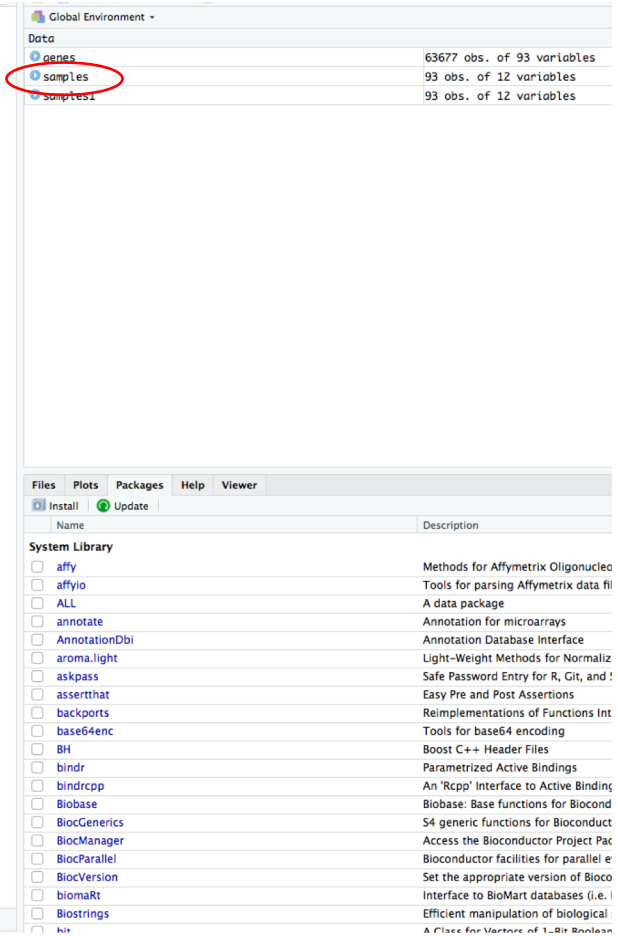


Correlations

- For example, the samples dataframe is shown below.

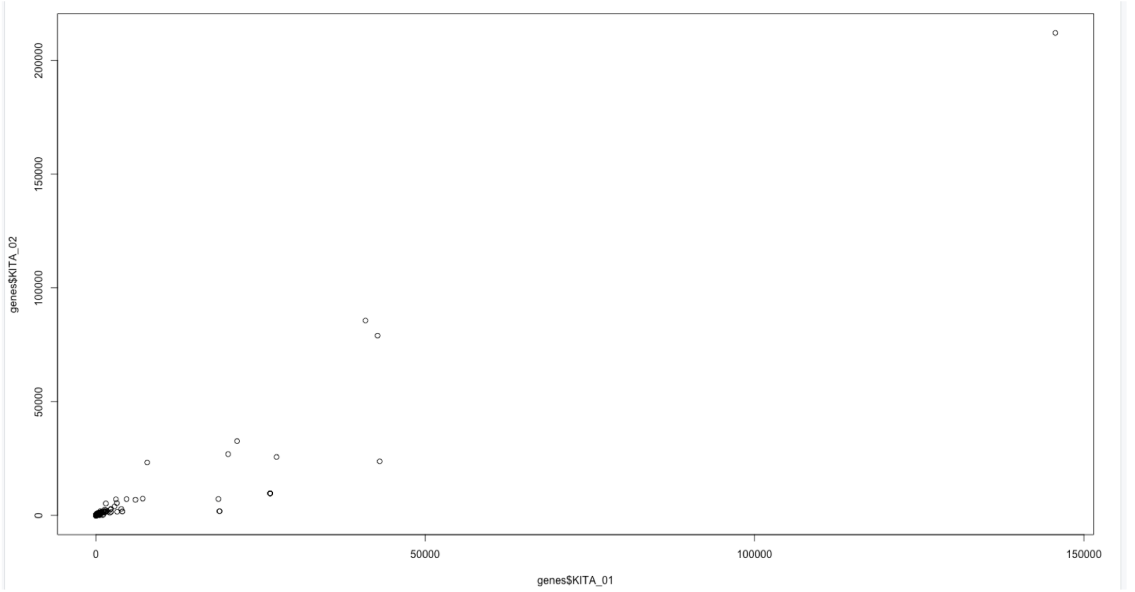
	patient	visit	reads	RIN	PF_BASES	PCT_RIBOSOMAL_BASES	PCT_CODING_BASES	PCT_UTR_BASES	PCT_INTRONIC_BASES	PCT_INTERGENIC_BASES	PCT_USABLE_BASES	Kit
KITA_01	701	Start	90426153	9.3	13202218338	0.124113	0.140400	0.426811	0.267503	0.042178	0.548038	A
KITA_02	701	END	267377263	8.6	39037080398	0.066727	0.142860	0.443182	0.304106	0.043625	0.568415	A
KITA_03	704	Start	78570128	8.0	11471238688	0.113600	0.136457	0.224241	0.400091	0.126288	0.345773	A
KITA_04	704	END	307503559	7.8	44895519614	0.066262	0.146475	0.296329	0.434821	0.056686	0.428476	A
KITA_05	705	Start	82847263	8.0	12095700398	0.046465	0.170637	0.260277	0.456688	0.066409	0.398425	A
KITA_06	705	END	274141940	8.4	40024723240	0.034185	0.157960	0.263815	0.486168	0.058190	0.407882	A
KITA_07	752	Start	86984040	8.0	12699669840	0.096885	0.145732	0.305159	0.399002	0.054372	0.437879	A
KITA_08	752	END	146987550	6.9	6405043214	0.061152	0.152052	0.275189	0.458097	0.054158	0.417983	A
KITA_09	758	Start	98781408	7.7	14422085568	0.057829	0.135968	0.249369	0.449739	0.107625	0.372103	A
KITA_10	758	END	183720498	7.2	9001741252	0.070695	0.137955	0.332468	0.409688	0.049877	0.460133	A
KITA_11	767	Start	95301342	8.6	13913995932	0.082855	0.157906	0.305779	0.402825	0.051459	0.449126	A
KITA_12	767	END	176832969	8.6	8535705748	0.024019	0.133970	0.257238	0.487987	0.096987	0.382289	A
KITA_13	802	Start	94216875	8.8	13755663750	0.053242	0.172447	0.352249	0.369567	0.052917	0.510840	A
KITA_14	802	END	122639695	8.5	5294507646	0.017898	0.117670	0.223497	0.478270	0.162812	0.327027	A
KITA_15	804	Start	94190477	8.8	13751809642	0.076057	0.154813	0.474294	0.254569	0.040691	0.611765	A
KITA_16	804	END	138092311	8.9	6744588990	0.018688	0.188472	0.295906	0.420230	0.076857	0.474282	A
KITA_17	805	Start	88272172	8.4	12887737112	0.128706	0.151487	0.444632	0.241227	0.035217	0.580059	A
KITA_18	805	END	93348304	8.4	4026135566	0.028936	0.164864	0.353051	0.395534	0.057968	0.503675	A
KITA_19	807	Start	93705311	7.9	13680975406	0.096365	0.155332	0.310041	0.380517	0.058853	0.449209	A
KITA_20	807	END	140934687	8.0	6662763290	0.026627	0.151086	0.304050	0.463772	0.054730	0.446190	A
KITA_21	81	Start	103838020	9.0	15160350920	0.057211	0.177771	0.362372	0.350822	0.052354	0.522998	A
KITA_22	81	END	186390542	9.0	8120433860	0.021239	0.197007	0.418132	0.315590	0.048232	0.601512	A
KITA_23	818	Start	91906677	8.0	13418374842	0.053674	0.148422	0.284098	0.433395	0.081063	0.415391	A
KITA_24	818	END	211406526	7.3	30865352796	0.106576	0.138381	0.281654	0.388132	0.086586	0.402503	A
KITA_25	84	Start	98329398	8.3	14356092108	0.025184	0.171931	0.315238	0.434677	0.053225	0.464829	A
KITA_26	84	END	87081380	5.9	12713881480	0.078224	0.141602	0.336391	0.395039	0.049685	0.465797	A
KITA_27	85	Start	79969417	9.9	11675534882	0.033660	0.166369	0.347731	0.369662	0.082839	0.485309	A
KITA_28	85	END	104804453	9.0	15301450138	0.056549	0.158241	0.329984	0.408404	0.047338	0.475575	A
KITA_29	85	END	109382253	7.3	5055557768	0.057913	0.153070	0.325247	0.409410	0.055016	0.462943	A
KITA_30	854	Start	95691589	7.5	13970971994	0.090277	0.136323	0.294689	0.373884	0.105877	0.412528	A
KITA_31	854	END	102913980	7.6	4824971208	0.045039	0.128817	0.395951	0.344423	0.086329	0.507408	A
KITA_32	869	Start	88316892	7.5	12894266232	0.056498	0.161979	0.438403	0.302592	0.041142	0.531772	A
KITA_33	869	END	298924437	7.4	43642967802	0.075273	0.150712	0.315831	0.403564	0.055248	0.450490	A
KITA_34	901	Start	87096253	9.7	12716052938	0.061019	0.177255	0.383252	0.330510	0.048587	0.539322	A
KITA_35	901	END	273283752	8.6	39899427792	0.085906	0.164199	0.313623	0.385754	0.051085	0.459320	A
KITA_36	907	Start	90062698	9.7	13149153908	0.068307	0.154121	0.440653	0.293203	0.044333	0.564456	A
KITA_37	907	Start	94496304	6.9	13796460384	0.030936	0.145616	0.311959	0.457591	0.054230	0.436070	A
KITA_38	907	END	300056915	8.8	43808309590	0.058681	0.137186	0.318260	0.397646	0.088769	0.430285	A
KITA_39	907	END	276088018	7.9	40308850628	0.071526	0.136063	0.332397	0.410377	0.050227	0.453739	A
KITA_40	908	END	88971434	6.6	12989829364	0.067821	0.132521	0.290038	0.448387	0.061847	0.411946	A
KITA_41	914	Start	87931046	9.7	12837932716	0.027555	0.189480	0.337202	0.393996	0.052029	0.500677	A
KITA_42	914	END	102836527	9.5	15014132942	0.081221	0.157961	0.376134	0.339047	0.046376	0.519058	A

Showing 1 to 42 of 93 entries



Correlations

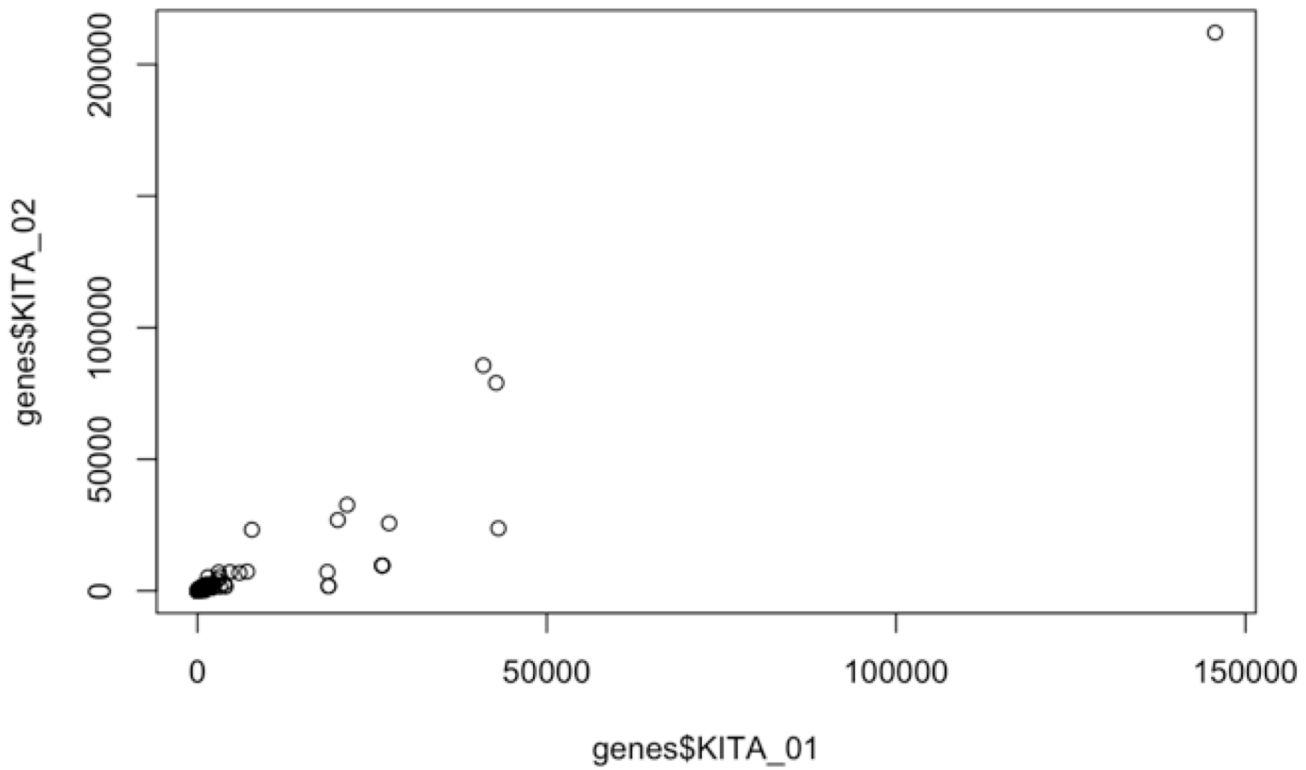
- One thing we notice is that the UID, or unique ID is a column name for genes, and within the first rows of samples. UID's are important concept and they are guaranteed unique row within a table, study, or in fact globally… or universally (UUID).
- Lots understand the data we are looking at a little better, and graph genes for two different samples by putting `plot(genes$KITA_01,genes$KITA_02)` into the console



Correlations

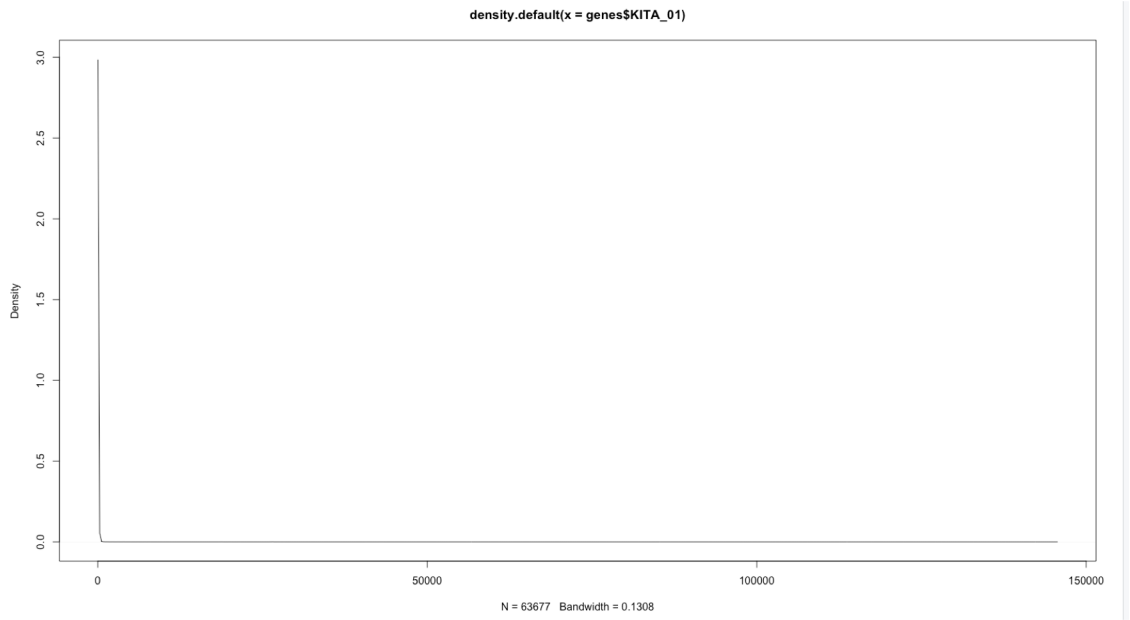
Ploting - looking at a little better

```
plot(genes$KITA_01,genes$KITA_02)
```



Correlations

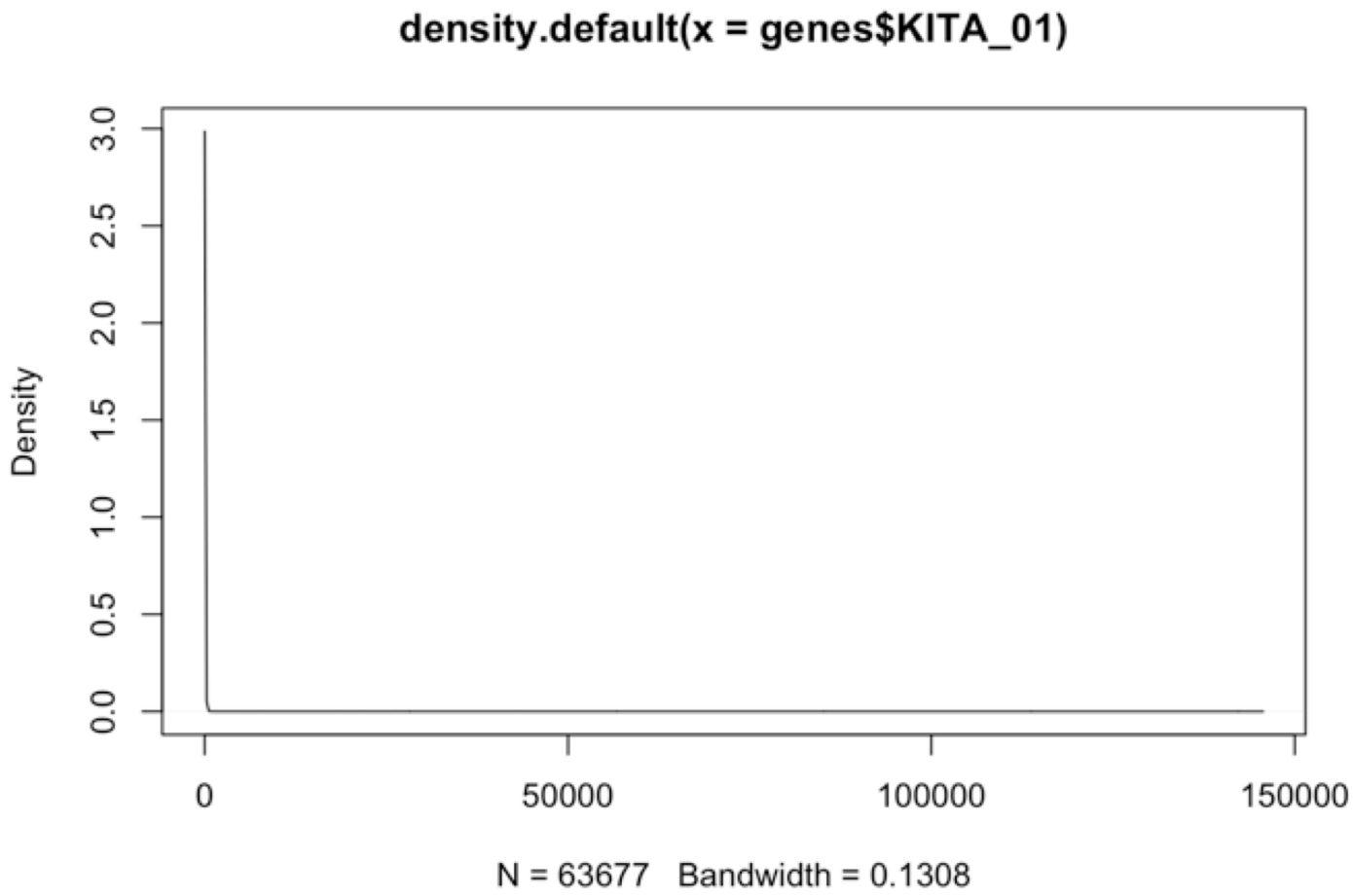
- While we graphed 63,677 different points for these two genes, you wouldn't know it as most of these have really very low values.
- Moreover, the data look correlated driven by large 4 points.
- This is an instance of what is likely non-parametric data (or data that doesn't fit a normal distribution. We can make a distribution of where the data are:



Correlations

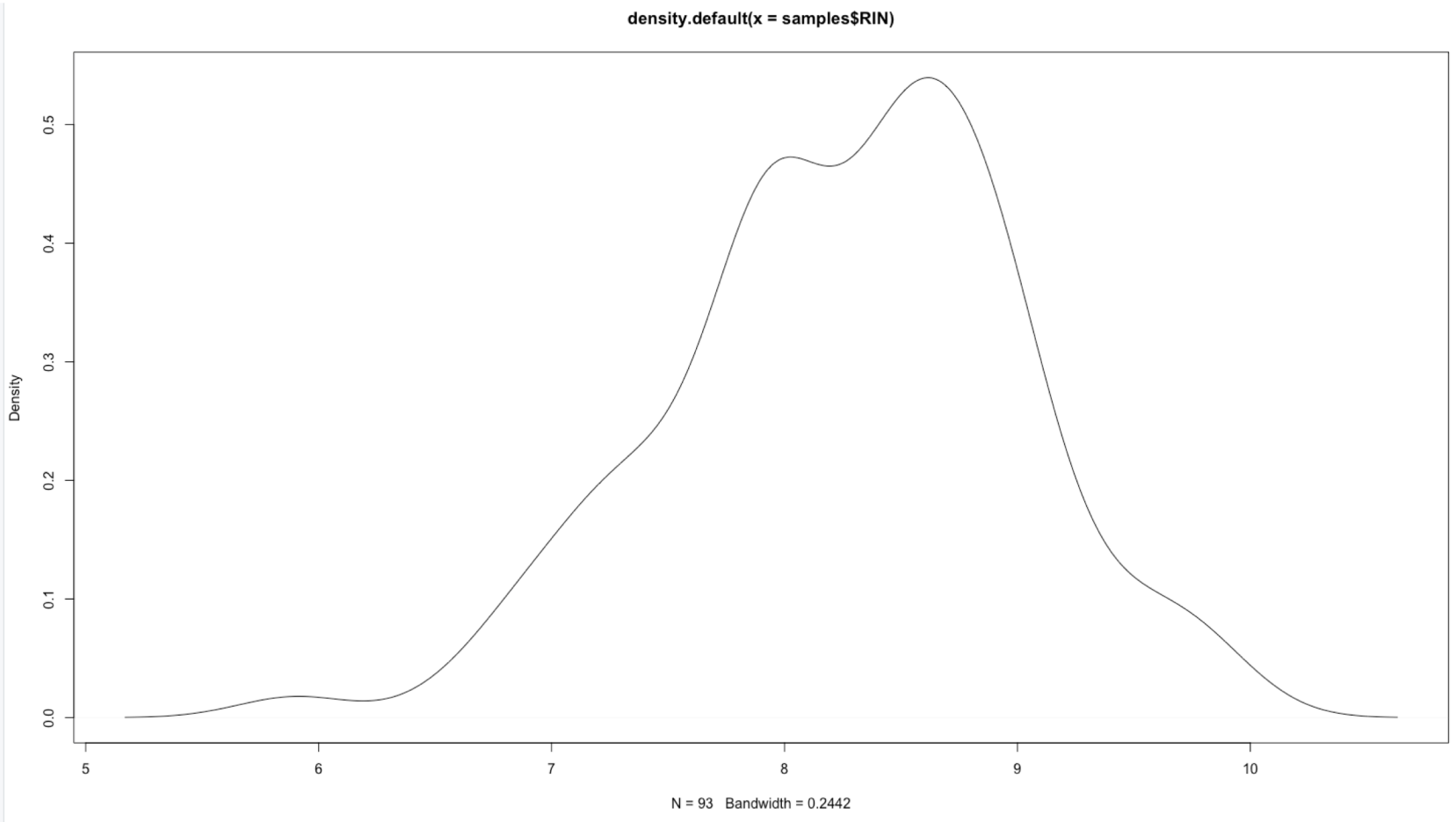
Making a distribution of where the data are

```
d <- density(genes$KITA_01) # returns the density data
plot(d) # plots the results
```



Correlations

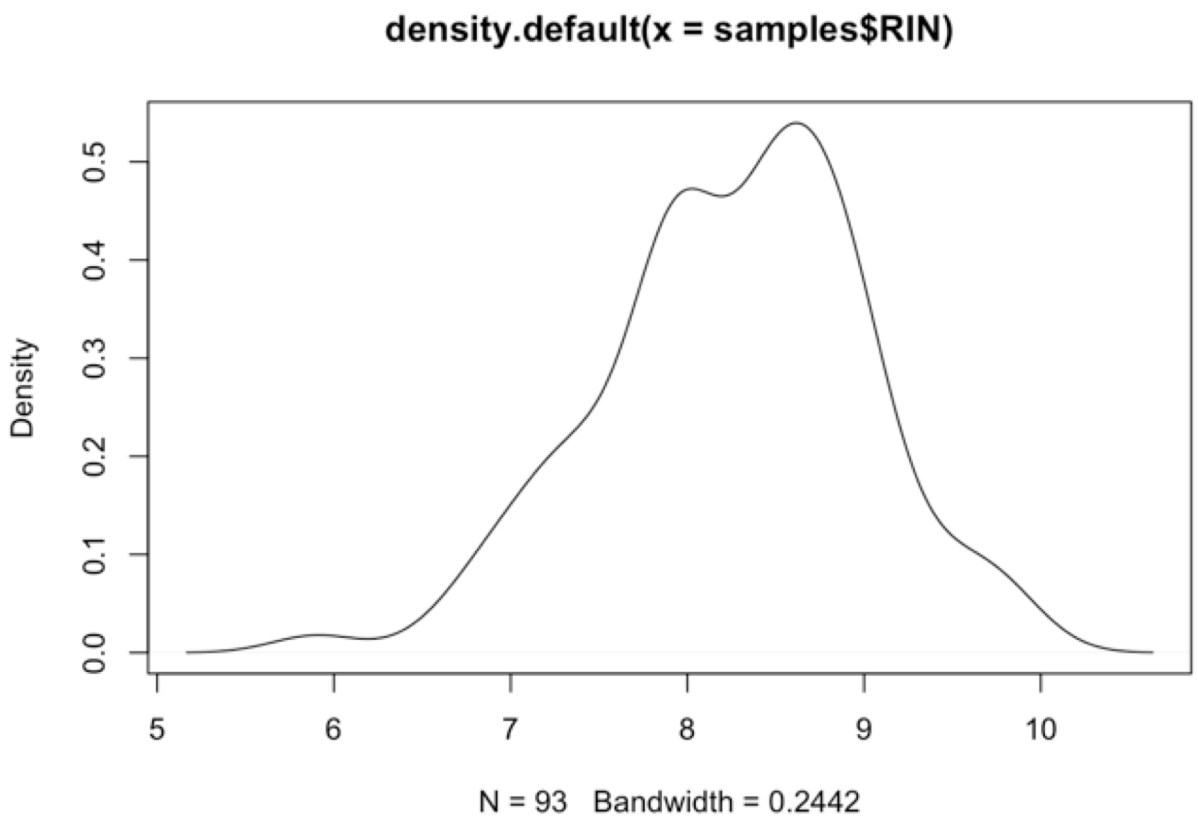
- Compare that to something more likely to be random normally distributed such as RIN for our samples:



Correlations

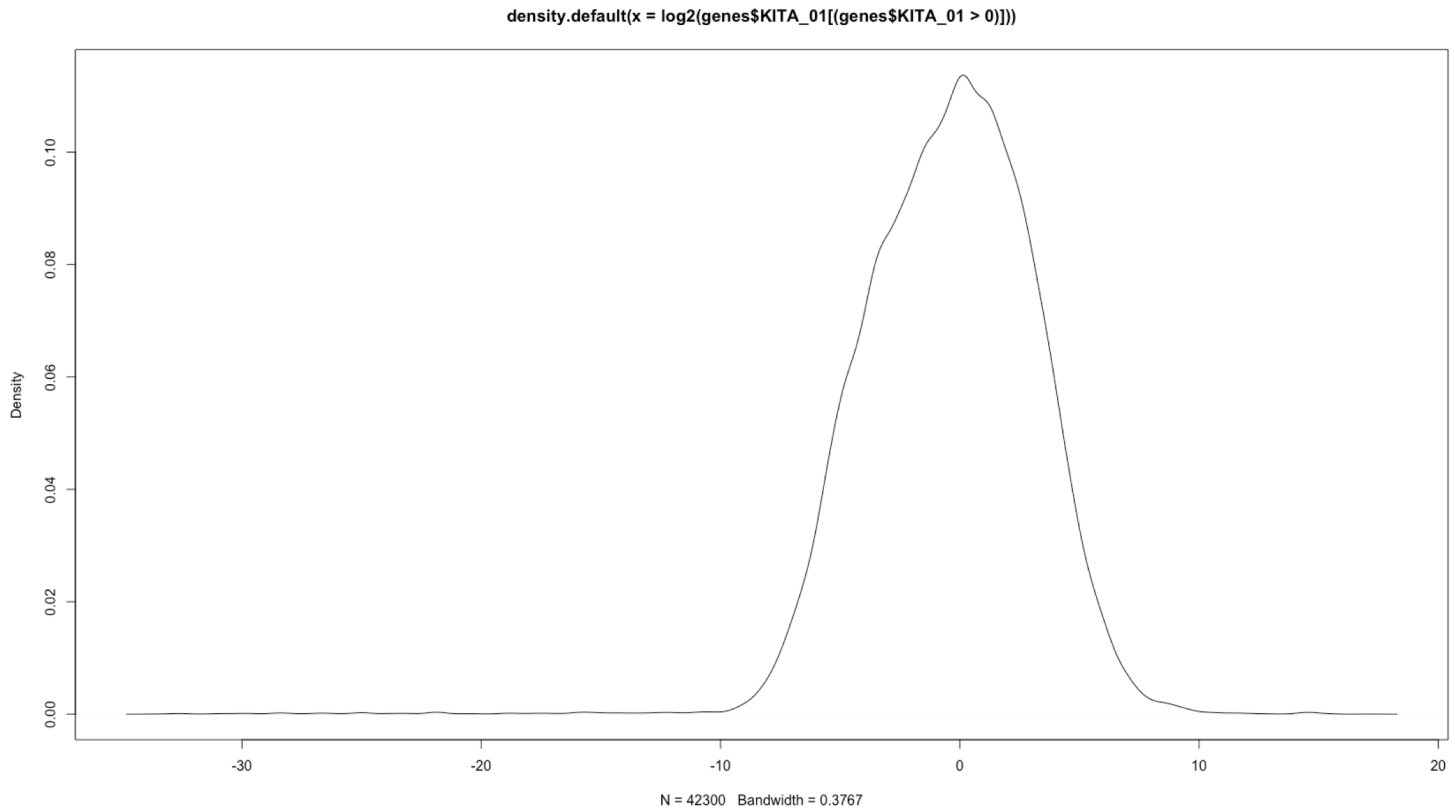
Comparing that to something more likely to be random normally distributed such as RIN for our samples:

```
dl <- density(samples$RIN) # returns the density data
plot(dl) # plots the results
```



Correlations

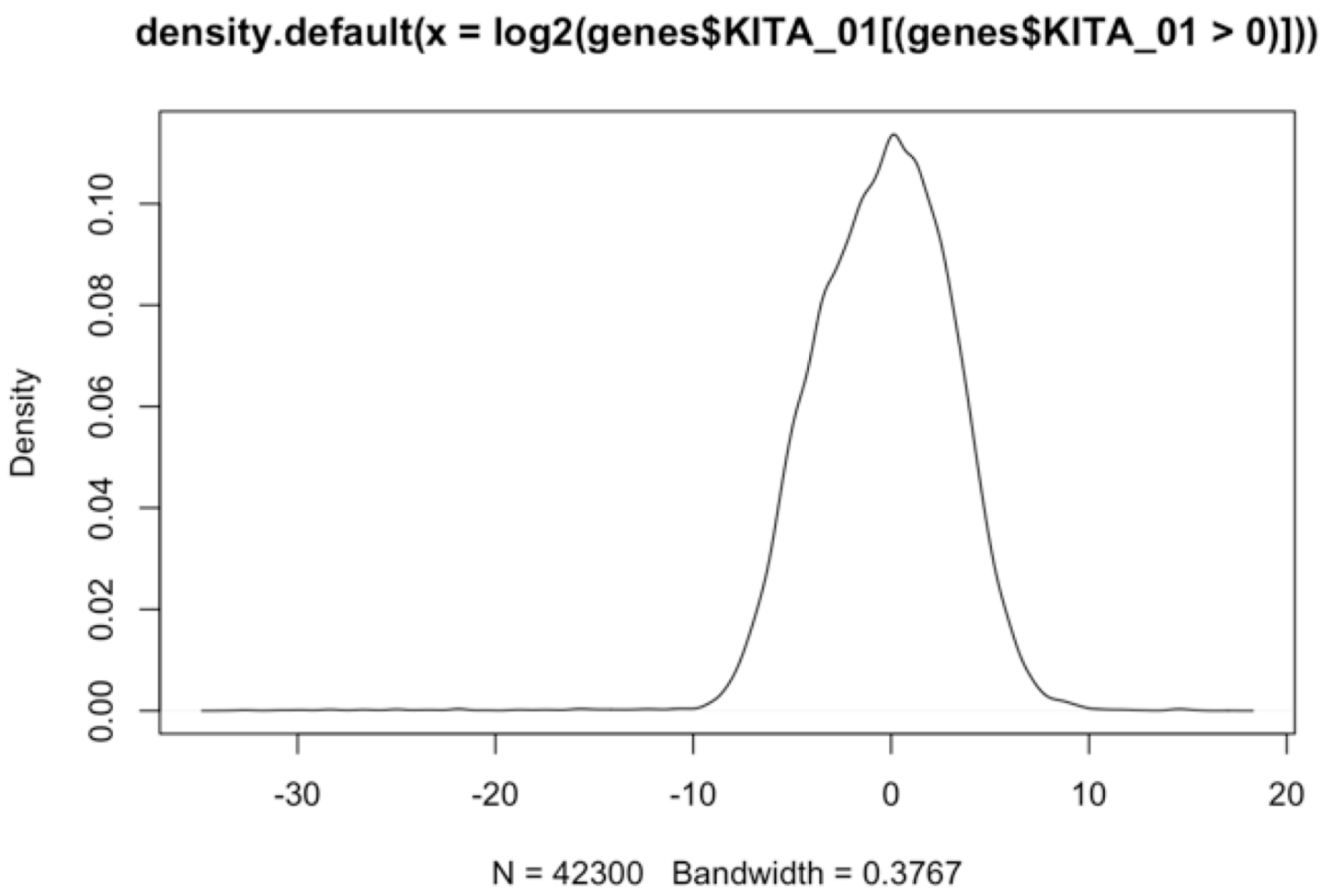
- Clearly we aren't normally distributed. Generally its wise to follow central-limit theorem when looking at data – or presume it to be non-parametric (Spearman), instead of parametric (Pearson).
- What we are doing is same as graphing on a log plot:



Correlations

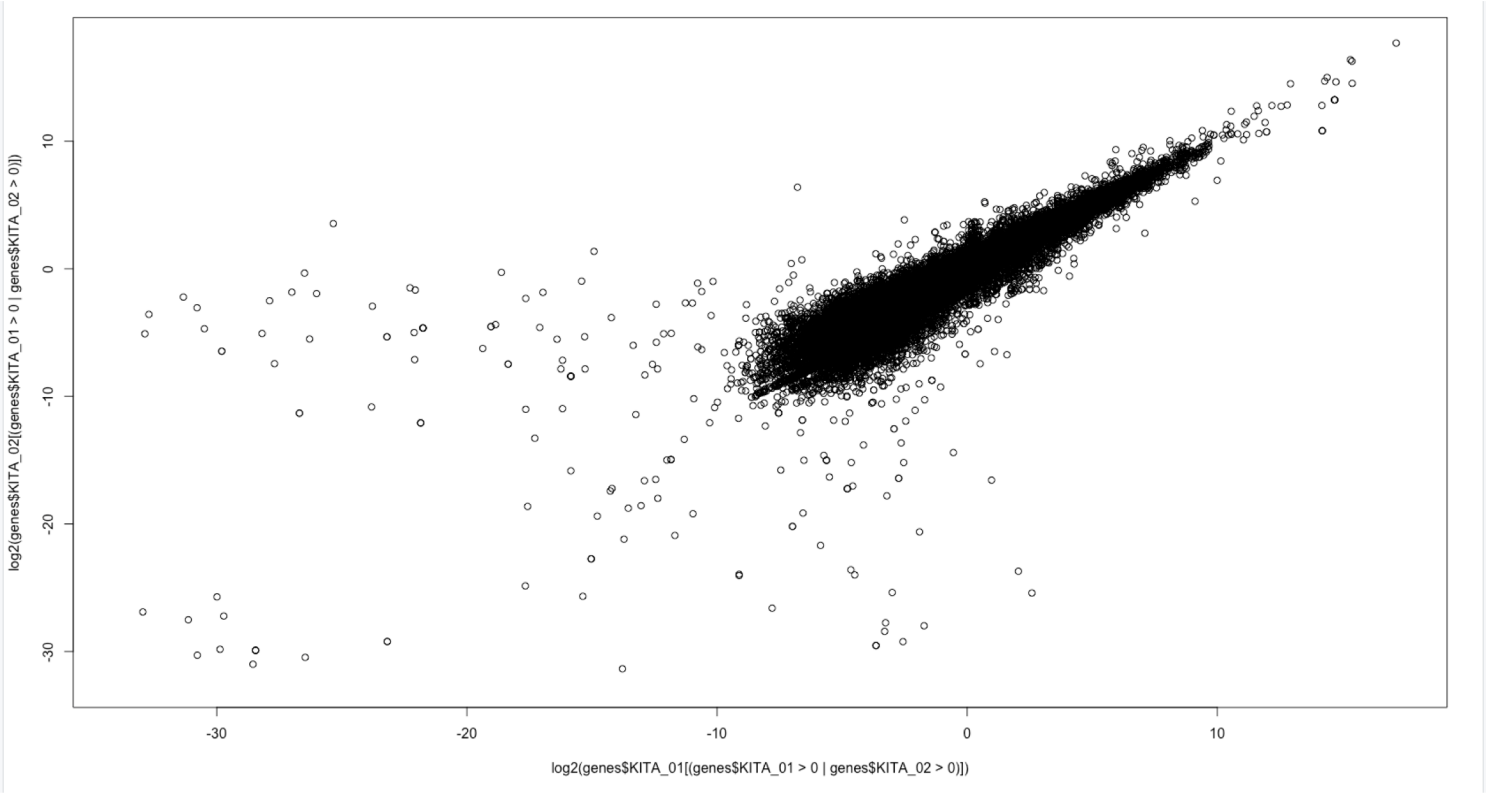
Graphing on a log plot

```
plot(density(log2(genes$KITA_01[(genes$KITA_01>0)])))
```



Correlations

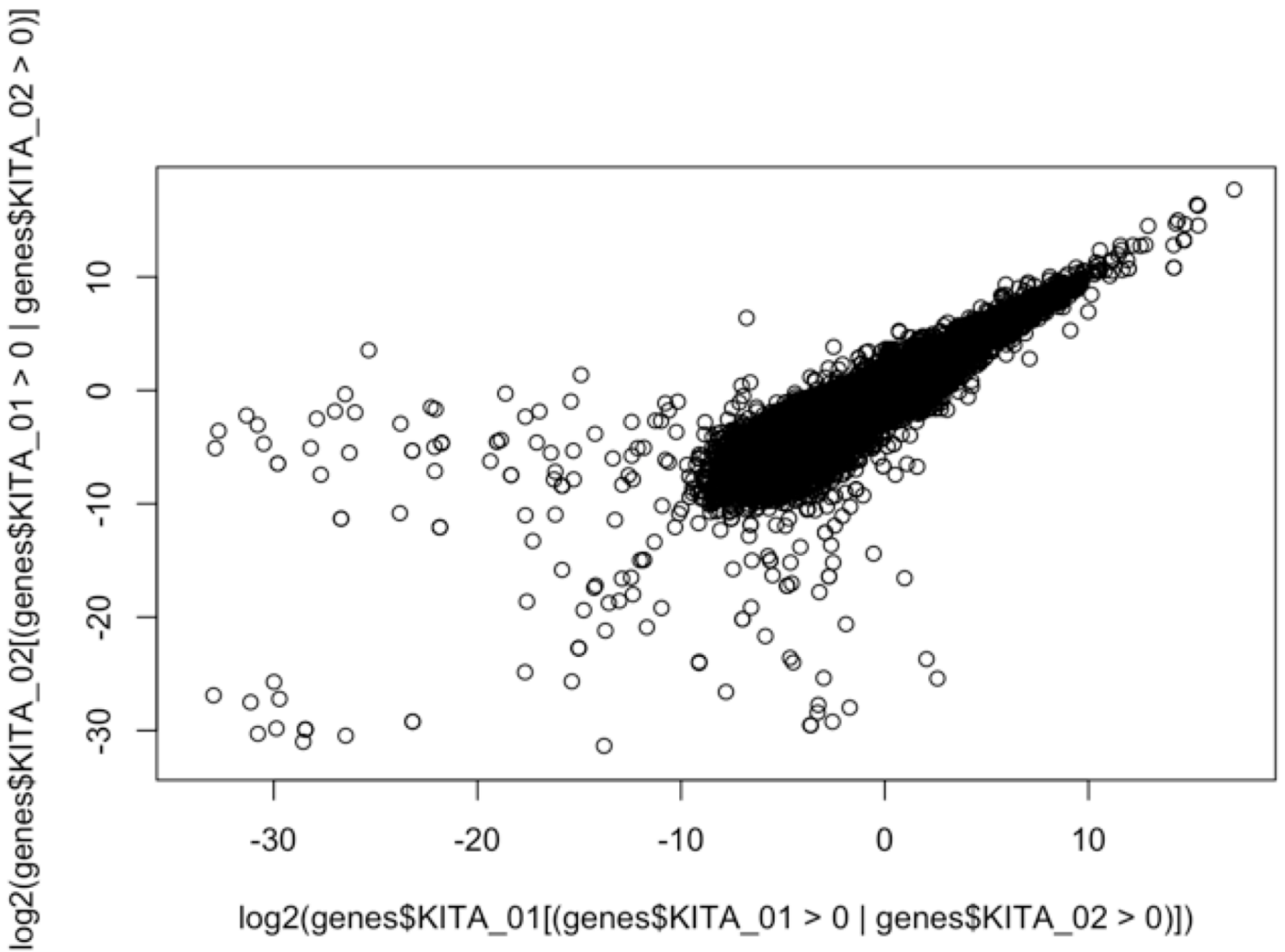
- Aren't log's fairly magical at making things normal? Note that we are not showing the zeroes by the filtering done in line (genes$KITA_01>0).
- Now lets actually plot the log2 values.



Correlations

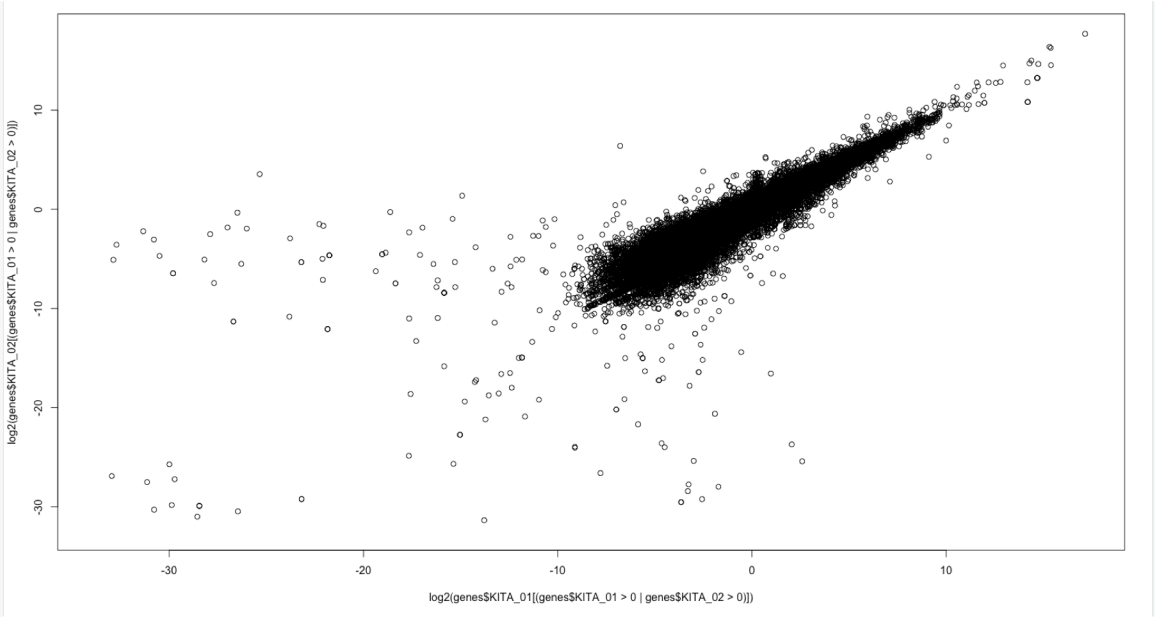
Ploting the log2 value

```
plot(log2(genes$KITA_01[(genes$KITA_01>0 |genes$KITA_02>0 )]),log2(genes$KITA_02[(genes$KITA_01>0 |genes$KITA_02>
0 )]))
```



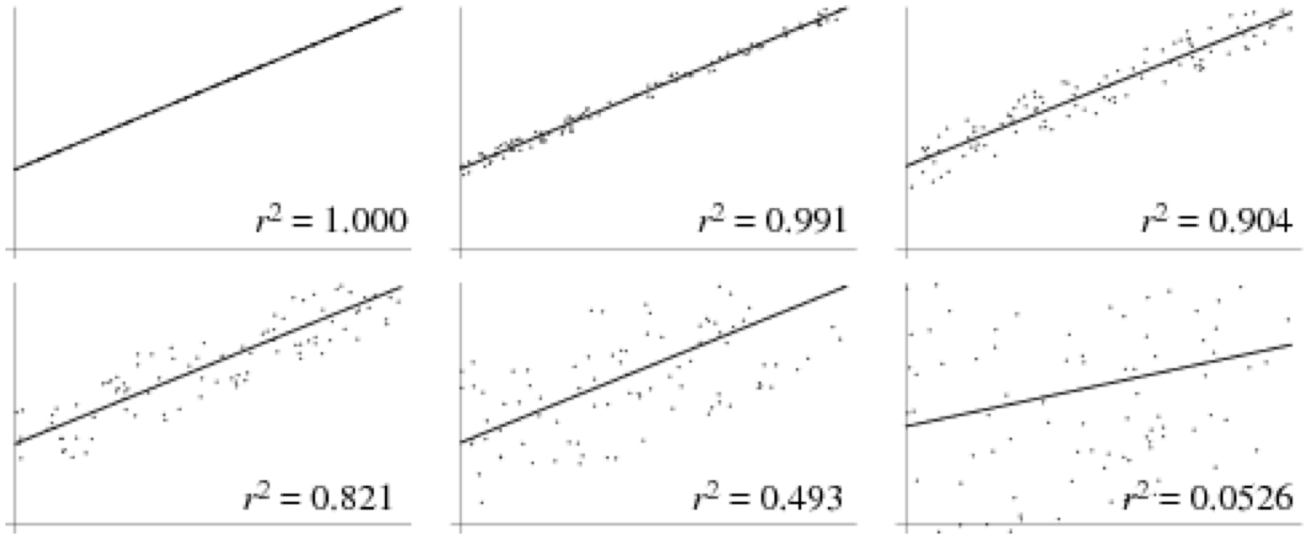
Correlations

- Now we see a plot showing us how two random different genes correlate. If we wanted, we could calculate a Spearman correlation or a Pearson correlation.
- A Pearson correlates the values, and the Spearman correlates the rank. Generally, Spearman is robust to the values and does not suffer from "tale wagging the dog".
- Calculating the Pearson correlation coefficient (r2) tell us about the percentage of variance given the other.



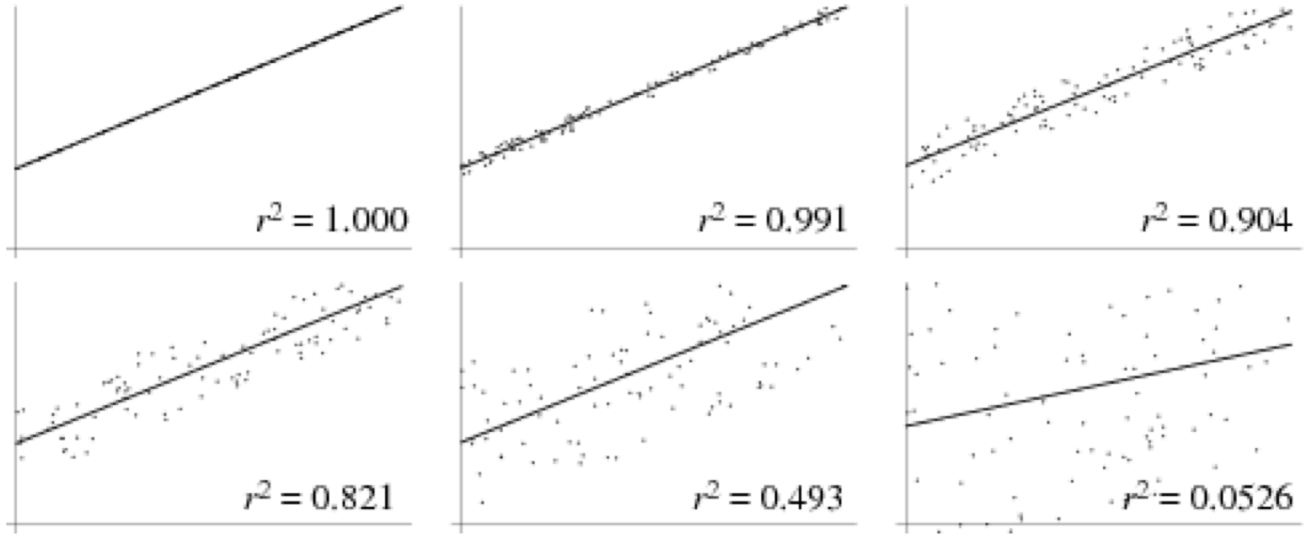
Correlations

- As we introduced above, correlation is generally a way to measure the strength of dependence of two variables on one another.
- By knowing on variable, how much do we know about another?
- One simple way to think about it as by percentage that one variable describes another variable.



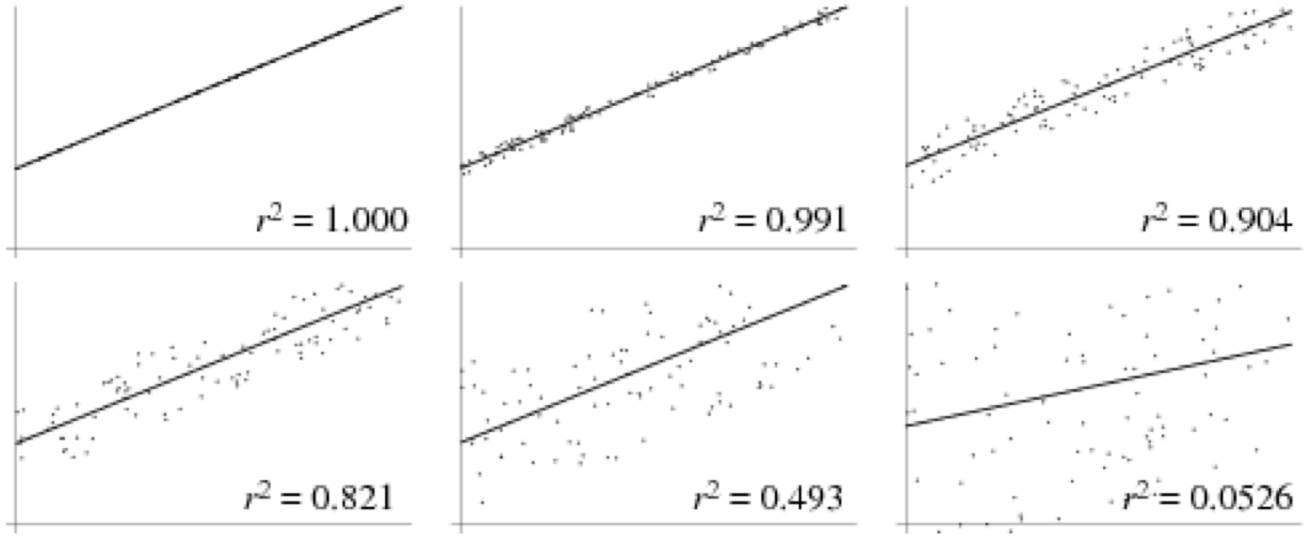
Correlations

- This is the correlation constant This is another way of thinking about r^2.
- R^2 is essentially the correlation coefficient.
- Sometimes, we just refer to *r,* without squaring, and thus this can go from -1 to 1 where -1 would be perfect inversely correlated.
- That is to say as one value goes up, another goes down.
- We are going to discuss this a few different ways, so do your best not to get hung up in the intricacies as that is not our goal in this course.



Correlations

- Also, as above there are many types of correlation coefficients that refer to correlating two vectors; or two samples.
- One type would be correlating the numbers as above, and this would be called a Pearson Correlation Constant.
- Another is to correlate their ranks, which provides a way to conduct correlation where the relative ranks of each value are correlated.



Correlations

- This is called a Spearman Correlation Constant. Spearman's are appropriate whenever there is concern that the data is not "normally distributed".
- At a high level, if one would make a histogram of the data it can provide us insight.
- In the dataset we are using to learn there are 93 samples.
- What if we took and measured the correlation constant between all samples?
- This would create a 93 by 93 matrix of correlation values.
- A function to help us with this would be "cor" in R, and we can apply it to genes.

````
72 ## Correlation function
73 ```{R}
74
75 corr<-cor(genes)
76 corr[1:5,1:5]
77
78 ```
````

Correlation function

```
corr<-cor(genes)
corr[1:5,1:5]
```

```
##           KITA_01   KITA_02   KITA_03   KITA_04   KITA_05
## KITA_01 1.0000000 0.9172325 0.7059474 0.7890462 0.8727615
## KITA_02 0.9172325 1.0000000 0.6796419 0.8972362 0.9245204
## KITA_03 0.7059474 0.6796419 1.0000000 0.7943030 0.7975012
## KITA_04 0.7890462 0.8972362 0.7943030 1.0000000 0.9774844
## KITA_05 0.8727615 0.9245204 0.7975012 0.9774844 1.0000000
```

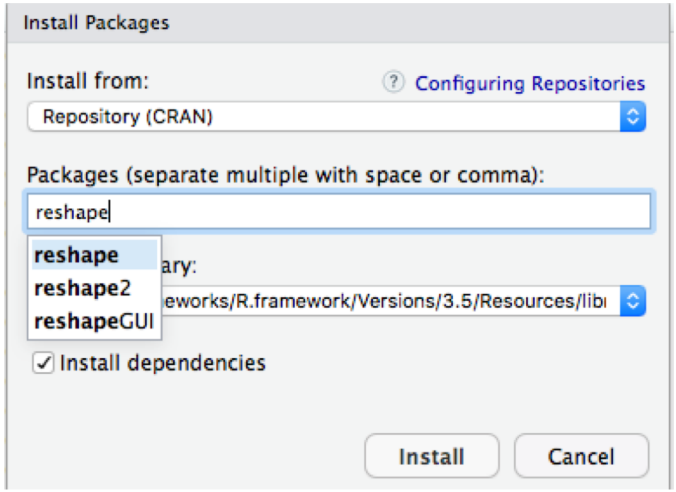
Correlations

- We can type `corr` to display it within the console, and we see something like this:

```
> corr<-cor(genes)
> corr
          KITA_01   KITA_02   KITA_03   KITA_04   KITA_05   KITA_06   KITA_07   KITA_08   KITA_09   KITA_10   KITA_11   KITA_12   KITA_13   KITA_14   KITA_15   KITA_16   KITA_17   KITA_18   KITA_19   KITA_20   KITA_21   KITA_22   KITA_23   KITA_24   KITA_25   KITA_26   KITA_27
KITA_01 1.0000000 0.9172325 0.7059474 0.7890462 0.8727615 0.7956106 0.9733134 0.5835039 0.9660088 0.5532845 0.9552791 0.5219514 0.9615811 0.9469390 0.9174128 0.5905192 0.9760214 0.9364722 0.9606003 0.5111776 0.9541974 0.6961002 0.9846005 0.8131689 0.9667256 0.9726216 0.9616185
KITA_02 0.9172325 1.0000000 0.6796419 0.8972362 0.9245204 0.9117325 0.9452424 0.7141847 0.9573101 0.6826740 0.9271196 0.6613212 0.9574658 0.9339753 0.9113776 0.7311093 0.9596523 0.9328599 0.9413480 0.6504227 0.9729275 0.8346086 0.9414465 0.9042873 0.9087120 0.9656873 0.8853493
KITA_03 0.7059474 0.6796419 1.0000000 0.7943030 0.7975012 0.7706625 0.7323131 0.7322714 0.7640362 0.7269355 0.7894807 0.7132718 0.5771759 0.5780154 0.6613543 0.7349513 0.6384038 0.5520092 0.7799926 0.7129279 0.6054180 0.7458627 0.6935726 0.7920496 0.6355836 0.6395789 0.6115812
KITA_04 0.7890462 0.8972362 0.7943030 1.0000000 0.9774844 0.9919876 0.8817409 0.9330817 0.9081349 0.9227431 0.9077163 0.9071693 0.8000657 0.7400728 0.6965319 0.9441305 0.7924033 0.7191202 0.9181642 0.9080819 0.8314789 0.9737039 0.8292602 0.9947637 0.7744088 0.8084763 0.7287852
KITA_05 0.8727615 0.9245204 0.7975012 0.9774844 1.0000000 0.9828198 0.9456743 0.8802853 0.9607031 0.8629674 0.9566094 0.8440385 0.8788957 0.8275283 0.7812882 0.8936671 0.8616558 0.8075502 0.9668399 0.8462579 0.8998106 0.9398359 0.9070623 0.9766745 0.8673822 0.8827022 0.8275389
KITA_06 0.7956106 0.9117325 0.7706625 0.9919876 0.9828198 1.0000000 0.8865923 0.9237530 0.9130027 0.9061886 0.9018744 0.8893546 0.8257012 0.7679088 0.7253954 0.9360298 0.8034567 0.7485785 0.9157446 0.8946749 0.8547839 0.9755647 0.8409616 0.9837911 0.7954117 0.8265169 0.7457619
KITA_07 0.9733134 0.9452424 0.7323131 0.8817409 0.9456743 0.8865923 1.0000000 0.7179584 0.9931976 0.6908351 0.9899104 0.6656300 0.9623132 0.9280620 0.8866171 0.7254739 0.9606474 0.9192896 0.9924982 0.6538992 0.9693629 0.8120728 0.9873080 0.8986229 0.9607828 0.9718299 0.9435834
KITA_08 0.5835039 0.7141847 0.7322714 0.9330817 0.8802853 0.9237530 0.7179584 1.0000000 0.7522318 0.9922794 0.7666746 0.9812828 0.5955596 0.5211426 0.4573534 0.9935530 0.5601423 0.4890810 0.7672814 0.9858355 0.6359819 0.9779325 0.6416408 0.9171242 0.5913599 0.5938364 0.5293333
KITA_09 0.9660088 0.9573101 0.7640362 0.9081349 0.9607031 0.9130027 0.9931976 0.7522318 1.0000000 0.7304422 0.9922388 0.7084825 0.9558163 0.9267786 0.8860939 0.7644747 0.9532858 0.9128925 0.9952049 0.6966333 0.9653128 0.8457108 0.9839626 0.9218518 0.9558838 0.9645551 0.9327773
KITA_10 0.5532845 0.6826740 0.7269355 0.9227431 0.8629674 0.9061886 0.6908351 0.9922794 0.7304422 1.0000000 0.7487411 0.9960447 0.5605917 0.4880463 0.4127655 0.9939076 0.5245767 0.4503985 0.7477827 0.9956604 0.6010443 0.9633588 0.6147626 0.9076500 0.5629960 0.5574831 0.4982348
          KITA_28   KITA_29   KITA_30   KITA_31   KITA_32   KITA_33   KITA_34   KITA_35   KITA_36   KITA_37   KITA_38   KITA_39   KITA_40   KITA_41   KITA_42   KITA_43   KITA_44   KITA_45   KITB_01   KITB_02   KITB_03   KITB_04   KITB_05   KITB_06   KITB_07   KITB_08   KITB_09
KITA_01 0.9791639 0.9507516 0.9881606 0.9399522 0.9568616 0.8602758 0.9818623 0.7984193 0.9325884 0.9522999 0.8716101 0.8928224 0.9824062 0.9675264 0.9534414 0.9410281 0.9712681 0.9631681 0.5282920 0.5480562 0.5877900 0.5718311 0.7145034 0.7039432 0.4661730 0.5575446 0.5090405
KITA_02 0.9615057 0.9517270 0.9396282 0.9018260 0.9038064 0.9540511 0.9187885 0.8998588 0.9185183 0.9459445 0.9742522 0.9761207 0.9567073 0.9096053 0.9583845 0.8391872 0.9711618 0.9666989 0.6511122 0.6642068 0.7168232 0.7046395 0.8165871 0.8296317 0.5925106 0.6791708 0.6416387
KITA_03 0.6789727 0.6318543 0.7263311 0.5290023 0.5817759 0.7761407 0.6672630 0.7963931 0.5253393 0.5825773 0.7475996 0.7588489 0.6636404 0.6717048 0.6054045 0.5420616 0.7473125 0.6157029 0.7539113 0.7597890 0.7448219 0.7583502 0.7135242 0.7667539 0.7155780 0.7081256 0.7329483
KITA_04 0.8286925 0.7888331 0.8512261 0.6715922 0.7128531 0.9848579 0.8041464 0.9973609 0.6802546 0.7562573 0.9698360 0.9672951 0.8093878 0.7928625 0.7629206 0.6296440 0.8900098 0.7937839 0.9137551 0.9195558 0.9390291 0.9396634 0.9392634 0.9723544 0.8771903 0.8994852 0.9082335
KITA_05 0.9018200 0.8618756 0.9241553 0.7678658 0.8045331 0.9823403 0.8908311 0.9816852 0.7709825 0.8393872 0.9754247 0.9767573 0.8899332 0.8825219 0.8369616 0.7464892 0.9463970 0.8691500 0.8457936 0.8559165 0.8879526 0.8812167 0.9197562 0.9343068 0.8100644 0.8561476 0.8433829
KITA_06 0.8432821 0.8055074 0.8580766 0.6966178 0.7295067 0.9828390 0.8188716 0.9932270 0.7087054 0.7834045 0.9778167 0.9719001 0.8280917 0.8127127 0.7836590 0.6561255 0.8978755 0.8169210 0.8898996 0.8987710 0.9310325 0.9247273 0.9463070 0.9666172 0.8618050 0.9041691 0.8924583
KITA_07 0.9834662 0.9486960 0.9951136 0.9030213 0.9291951 0.9323701 0.9822318 0.8915228 0.8975079 0.9383939 0.9361456 0.9496587 0.9777762 0.9605062 0.9384314 0.8924398 0.9916388 0.9595674 0.6565123 0.6720731 0.7170435 0.7011325 0.8204622 0.8118359 0.5963938 0.6749322 0.6433796
KITA_08 0.6255820 0.5733074 0.6735660 0.4278141 0.4842463 0.8684739 0.6238020 0.9319766 0.4285180 0.5383994 0.8417973 0.8330723 0.6023402 0.6149298 0.5256206 0.4111711 0.7107660 0.5747157 0.9710908 0.9743642 0.9908743 0.9834481 0.9504914 0.9708217 0.9488659 0.9476374 0.9765632
KITA_09 0.9781933 0.9454285 0.9897357 0.8928328 0.9203689 0.9514998 0.9712894 0.9140188 0.8864087 0.9344092 0.9543238 0.9652864 0.9718589 0.9607508 0.9338609 0.8804052 0.9934949 0.9542470 0.6971991 0.7097264 0.7517828 0.7410234 0.8420749 0.8430541 0.6386898 0.7074760 0.6851313
KITA_10 0.5916973 0.5416127 0.6463049 0.3895431 0.4525685 0.8506907 0.5911544 0.9170303 0.3859935 0.5026785 0.8177392 0.8091625 0.5683729 0.5891557 0.4866609 0.3784470 0.6825933 0.5362237 0.9802924 0.9766757 0.9806351 0.9855026 0.9218554 0.9517933 0.9565155 0.9269178 0.9827134
          KITB_10   KITB_11   KITB_12   KITB_13   KITB_14   KITB_15   KITB_16   KITB_17   KITB_18   KITB_19   KITB_20   KITB_21   KITB_22   KITB_23   KITB_24   KITC_01   KITC_02   KITC_03   KITC_04   KITC_05   KITC_06   KITC_07   KITC_08   KITC_09   KITC_10   KITC_11   KITC_12
KITA_01 0.6124591 0.6750295 0.6297774 0.5373403 0.6222078 0.6516379 0.5345814 0.5646314 0.5642766 0.5539129 0.6029237 0.6111609 0.5223321 0.7356004 0.5433894 0.5669332 0.6369465 0.5693011 0.5475406 0.4626927 0.6373221 0.5732982 0.6692924 0.6311313 0.7046560 0.5796642 0.6996117
KITA_02 0.7417355 0.8053520 0.7708376 0.6567508 0.7492143 0.7645462 0.5950664 0.6898210 0.6830722 0.6840046 0.7448084 0.7477869 0.6528520 0.8656129 0.6747673 0.7052486 0.7709038 0.7051659 0.6732492 0.5939712 0.7907647 0.7280412 0.8143308 0.7834660 0.8507966 0.7284521 0.8491231
KITA_03 0.7583357 0.7657049 0.7436701 0.7568071 0.7709130 0.7954772 0.8257844 0.7590851 0.7521113 0.7493853 0.7571740 0.7517584 0.7331815 0.7379933 0.7416544 0.7603854 0.6898279 0.7202719 0.7658548 0.7298063 0.7421466 0.7307255 0.7477422 0.7325194 0.7036074 0.7485435 0.7313745
KITA_04 0.9538630 0.9714958 0.9583929 0.9171795 0.9567687 0.9571678 0.8608458 0.9325124 0.9289707 0.9295368 0.9529793 0.9540783 0.9143113 0.9674661 0.9248884 0.9366288 0.8827881 0.9117002 0.9220154 0.8831421 0.9556418 0.9361013 0.9577152 0.9479145 0.9174394 0.9439013 0.9516915
KITA_05 0.9040128 0.9316856 0.9135447 0.8514341 0.9029917 0.9051105 0.7966407 0.8729999 0.8704527 0.8687488 0.9020606 0.9048462 0.8526287 0.9441723 0.8655004 0.8727247 0.8530873 0.8641840 0.8531947 0.8078891 0.9089306 0.8804948 0.9161485 0.9031121 0.8816817 0.8870539 0.9164947
KITA_06 0.9443822 0.9682219 0.9541506 0.8936955 0.9424646 0.9400606 0.8264397 0.9167327 0.9126983 0.9144117 0.9461479 0.9480086 0.8996818 0.9722632 0.9116101 0.9157971 0.8790120 0.9111654 0.8944478 0.8576195 0.9494621 0.9286081 0.9524014 0.9454658 0.9053902 0.9324437 0.9492135
KITA_07 0.7370999 0.7884105 0.7558183 0.6662109 0.7425500 0.7633672 0.6367430 0.6904785 0.6920030 0.6830954 0.7264395 0.7353480 0.6587593 0.8401004 0.6759428 0.6955334 0.7361347 0.6868378 0.6744396 0.6004747 0.7583003 0.7025370 0.7830891 0.7508913 0.8031759 0.7082383 0.8028216
KITA_08 0.9877440 0.9805084 0.9860400 0.9744802 0.9871265 0.9769076 0.9191569 0.9805522 0.9815513 0.9830710 0.9756790 0.9866451 0.9824427 0.9499752 0.9806200 0.9752449 0.8466538 0.9412201 0.9683858 0.9601333 0.9625614 0.9699261 0.9595697 0.9600849 0.8737250 0.9730416 0.9174800
KITA_09 0.7744367 0.8217650 0.7949961 0.7055346 0.7796789 0.7962285 0.6710790 0.7297323 0.7304937 0.7231677 0.7677615 0.7743121 0.6992495 0.8696791 0.7163150 0.7369757 0.7766897 0.7230876 0.7135728 0.6433861 0.7990641 0.7466860 0.8169076 0.7898041 0.8409171 0.7512557 0.8415865
KITA_10 0.9836639 0.9646274 0.9802917 0.9821098 0.9828678 0.9666703 0.9256143 0.9817160 0.9870482 0.9853227 0.9746064 0.9816227 0.9882819 0.9250585 0.9843137 0.9796023 0.8410593 0.9234630 0.9735506 0.9733119 0.9549738 0.9686555 0.9387317 0.9456385 0.8668222 0.9751164 0.9048380
          KITC_13   KITC_14   KITC_15   KITC_16   KITC_17   KITC_18   KITC_19   KITC_20   KITC_21   KITC_22   KITC_23   KITC_24
KITA_01 0.5245066 0.5830744 0.5295109 0.4948320 0.4859720 0.5382275 0.5654174 0.5649876 0.6908371 0.6295027 0.6609623 0.7005980
KITA_02 0.6670238 0.7207490 0.6642254 0.6087756 0.5879450 0.6909019 0.7173711 0.7136423 0.8370392 0.7767490 0.8117869 0.8507139
KITA_03 0.7031301 0.7652874 0.7509054 0.7542432 0.7585470 0.7326570 0.7430528 0.7341356 0.7274116 0.7590020 0.7487184 0.7397633
KITA_04 0.8851601 0.9432183 0.9192101 0.8889428 0.8727748 0.9262210 0.9376470 0.9323872 0.9484329 0.9609376 0.9618602 0.9613430
KITA_05 0.8255240 0.8815064 0.8535801 0.8155049 0.7945905 0.8633623 0.8768585 0.8764409 0.9170711 0.9109164 0.9180238 0.9276069
KITA_06 0.8685812 0.9234169 0.9002733 0.8591603 0.8350461 0.9122332 0.9225142 0.9217959 0.9485204 0.9512543 0.9553681 0.9597216
KITA_07 0.6526953 0.7088717 0.6595789 0.6202197 0.6051531 0.6743315 0.6981885 0.6986974 0.8005864 0.7528181 0.7775019 0.8082940
KITA_08 0.9287530 0.9730194 0.9681087 0.9481638 0.9297259 0.9741967 0.9742030 0.9704684 0.9204335 0.9761026 0.9541386 0.9280051
KITA_09 0.6962017 0.7498914 0.7012196 0.6588991 0.6432362 0.7196916 0.7429901 0.7400755 0.8349197 0.7912251 0.8173154 0.8457766
KITA_10 0.9372861 0.9767098 0.9740396 0.9566766 0.9431472 0.9819306 0.9817211 0.9694055 0.9028585 0.9686187 0.9470557 0.9159120
 [ reached getOption("max.print") -- omitted 83 rows ]
```

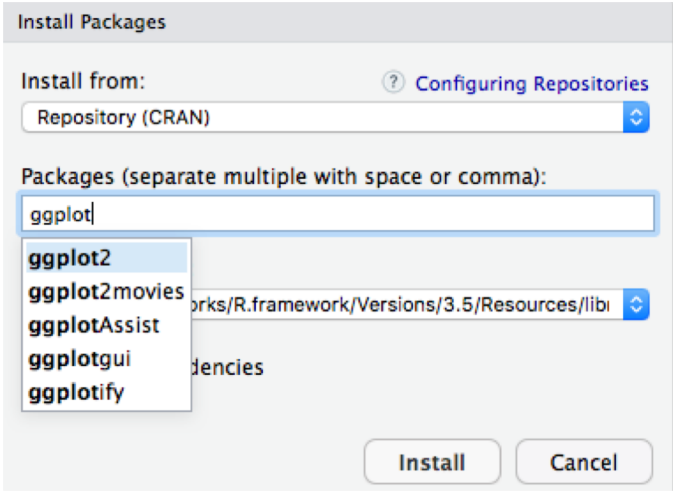
Correlations

- We note that the obviously when one variable is perfectly correlated to itself, just as we would expect.
- Now we can't see the matrix easily, and honestly they are difficult to compute.
- We could reshape this as a list of pairs and that would be easier.
- This is core to the concept of "melt", which turns a square matrix, into minimal (in this case pairwise) components.
- The "melt" function is in the package "reshape" thus its prior installation is required.



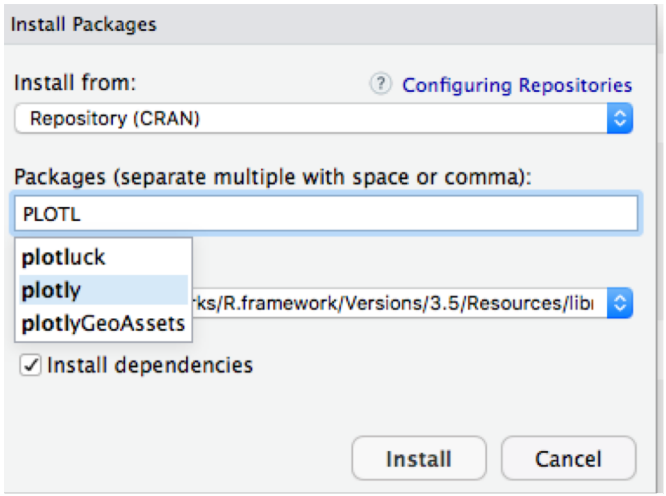
Correlations

- Also “ggplot2” package and is required for the following code.



Correlations

- Lets use a different library, “plotly” which adds additional functionality to ggplot.
- You may need to install the plotly packages via *install.packages(‘plotly’)* and some individuals may be requested to install development version of ggplot, you can do this via: *install_github(“ggplot2”, “hadley”, “develop”).*
- Here we remove some labels and make it work with hovering.
- The entire block is:



Correlations

Reshaping this as a list of pairs

```
#install.packages("reshape2")
#install.packages("ggplot2")

library(ggplot2)
library(reshape2)
library(plotly)
```

```
##
## Attaching package: 'plotly'
```

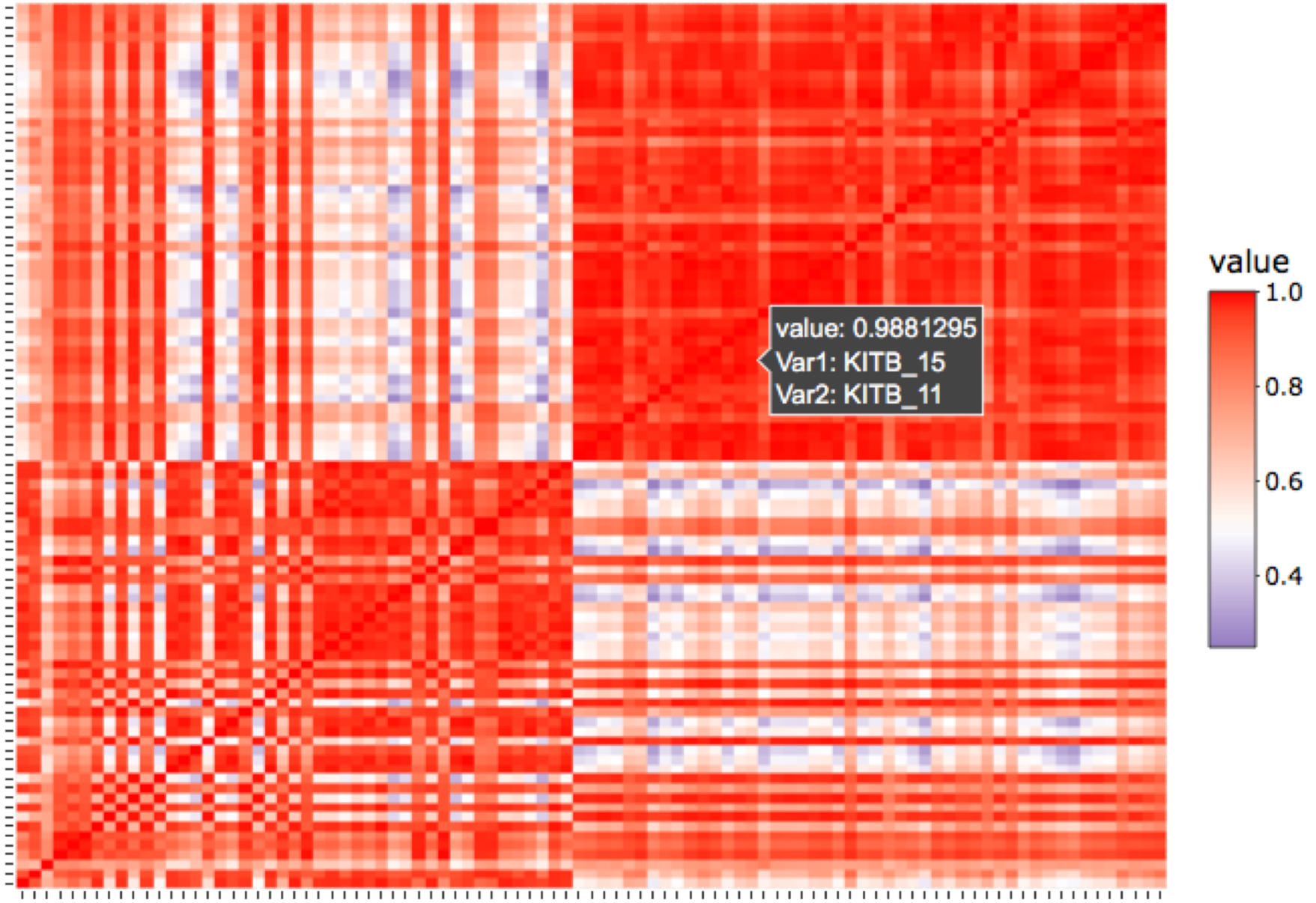
```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
corr<-cor(genes)
melted_corr <- melt(corr)
p<-ggplot(melted_corr , aes(x = Var1, y = Var2)) + geom_raster(aes(fill = value)) + scale_fill_gradient2(low="nav
y", mid="white", high="red", midpoint=0.5) + theme( plot.title = element_blank(),axis.text.x = element_blank(), a
xis.text.y = element_blank(), axis.title.y = element_blank(), axis.title.x = element_blank())
ggplotly(p)
```

Correlations



R Markdown Week 5 Lecture 1

~/Documents/R_working_directory/Rmarkdown_Week_5_Lecture_1.html

TRGN599_Week_5_Lecture_1

Enrique I. Velazquez Villarreal, MD, PhD, MPH, MS

2/1/2019

Correlations analysis

```
# Check your current working directory
getwd()
```

```
## [1] "/Users/enriquevelazquez/Documents/R_working_directory"
```

```
# Set your working directory
setwd("/Users/enriquevelazquez/Documents/R_working_directory")
```

Uploading datasets

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

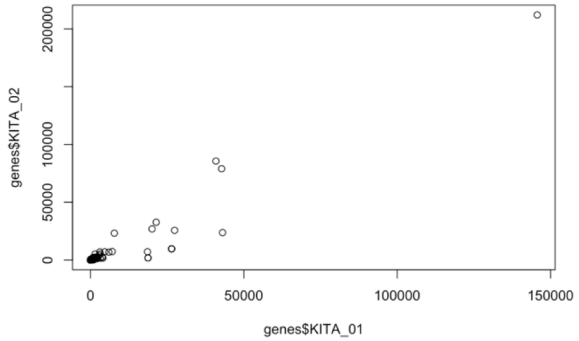
```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
samples <- read.csv('sample_info.csv', header = TRUE, sep = ",", quote = "'", dec = ".", fill = TRUE, row.names =
1)

genes <- read.csv('expression_results.csv',header = TRUE, sep = ",", quote = "'", dec = ".", fill = TRUE, row.nam
es = 1)
```

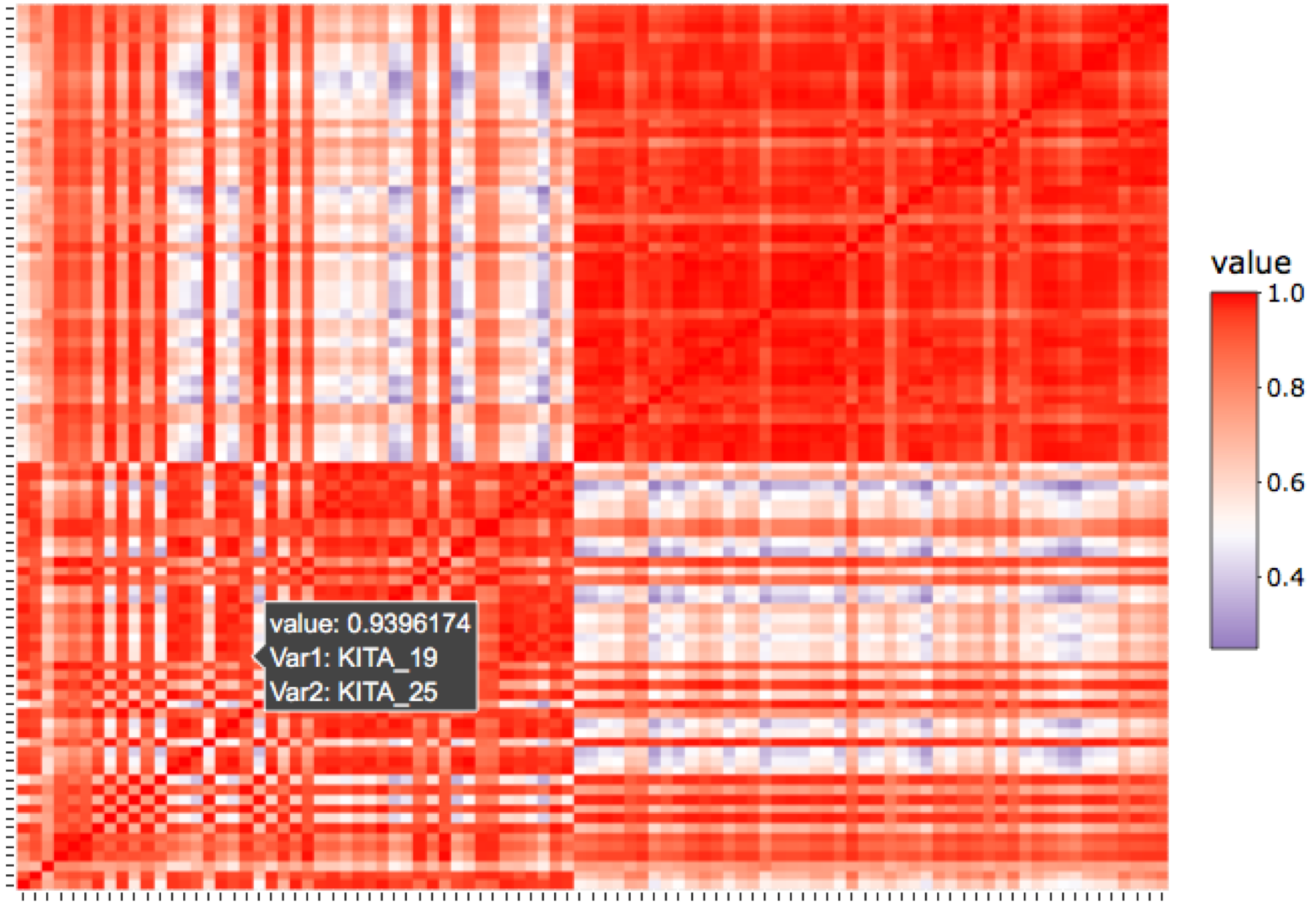
Ploting - looking at a little better

```
plot(genes$KITA_01,genes$KITA_02)
```



Making a distribution of where the data are

Correlations



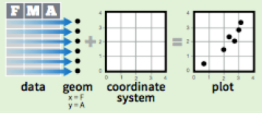
GGPLOT2

Data Visualization with ggplot2 Cheat Sheet

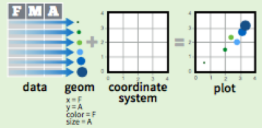
RStudio

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same few components: a **data** set, a set of **geoms**—visual marks that represent data points, and a **coordinate system.**



To display data values, map variables in the data set to aesthetic properties of the geom like **size**, **color**, and **x** and **y** locations.



Build a graph with **qplot()** or **ggplot()**

aesthetic mappings · data · geom

qplot(x = cty, y = hwy, color = cyl, data = mpg, geom = "point"**)**
Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

ggplot(data = mpg, **aes(**x = cty, y = hwy**))**
Begins a plot that you finish by adding layers to. No defaults, but provides more control than qplot().

data

```
ggplot(mpg, aes(hwy, cty)) +
 geom_point(aes(color = cyl)) +
 geom_smooth(method ="lm") +
 coord_cartesian() +
 scale_color_gradient() +
 theme_bw()
```

add layers, elements with +

layer = geom + default stat + layer specific mappings

additional elements

Add a new layer to a plot with a **geom_*()** or **stat_*()** function. Each provides a geom, a set of aesthetic mappings, and a default stat and position adjustment.

last_plot()
Returns the last plot

ggsave("plot.png", width = 5, height = 5)
Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

Geoms - Use a geom to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

One Variable

Continuous

a <- ggplot(mpg, aes(hwy))

a + geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size
b + geom_area(aes(y = ..density..), stat = "bin")

a + geom_density(kernel = "gaussian"**)**
x, y, alpha, color, fill, linetype, size, weight
b + geom_density(aes(y = ..county..))

a + geom_dotplot()
x, y, alpha, color, fill

a + geom_freqpoly()
x, y, alpha, color, linetype, size
b + geom_freqpoly(aes(y = ..density..))

a + geom_histogram(binwidth = 5**)**
x, y, alpha, color, fill, linetype, size, weight
b + geom_histogram(aes(y = ..density..))

Discrete

b <- ggplot(mpg, aes(fl))

b + geom_bar()
x, alpha, color, fill, linetype, size, weight

Graphical Primitives

c <- ggplot(map, aes(long, lat))

c + geom_polygon(aes(group = group)**)**
x, y, alpha, color, fill, linetype, size

d <- ggplot(economics, aes(date, unemploy))

d + geom_path(lineend="butt", linejoin="round', linemitre=1**)**
x, y, alpha, color, linetype, size

d + geom_ribbon(aes(ymin=unemploy - 900, ymax=unemploy + 900**)**
x, ymax, ymin, alpha, color, fill, linetype, size

e <- ggplot(seals, aes(x = long, y = lat))

e + geom_segment(aes(
xend = long + delta_long,
yend = lat + delta_lat**)**
x, xend, y, yend, alpha, color, linetype, size

e + geom_rect(aes(xmin = long, ymin = lat,
xmax= long + delta_long,
ymax = lat + delta_lat**)**
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

Two Variables

Continuous X, Continuous Y

f <- ggplot(mpg, aes(cty, hwy))

f + geom_blank()

f + geom_jitter()
x, y, alpha, color, fill, shape, size

f + geom_point()
x, y, alpha, color, fill, shape, size

f + geom_quantile()
x, y, alpha, color, linetype, size, weight

f + geom_rug(sides = "bl"**)**
alpha, color, linetype, size

f + geom_smooth(model = lm**)**
x, y, alpha, color, fill, linetype, size, weight

f + geom_text(aes(label = cty**))**
x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

Discrete X, Continuous Y

g <- ggplot(mpg, aes(class, hwy))

g + geom_bar(stat = "identity")
x, y, alpha, color, fill, linetype, size, weight

g + geom_boxplot()
lower, middle, upper, x, ymax, ymin, alpha, color, fill, linetype, shape, size, weight

g + geom_dotplot(binaxis = "y", stackdir = "center"**)**
x, y, alpha, color, fill

g + geom_violin(scale = "area"**)**
x, y, alpha, color, fill, linetype, size, weight

Discrete X, Discrete Y

h <- ggplot(diamonds, aes(cut, color))

h + geom_jitter()
x, y, alpha, color, fill, shape, size

Continuous Bivariate Distribution

i <- ggplot(movies, aes(year, rating))

i + geom_bin2d(binwidth = c(5, 0.5)**)**
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size, weight

i + geom_density2d()
x, y, alpha, colour, linetype, size

i + geom_hex()
x, y, alpha, colour, fill size

Continuous Function

j <- ggplot(economics, aes(date, unemploy))

j + geom_area()
x, y, alpha, color, fill, linetype, size

j + geom_line()
x, y, alpha, color, linetype, size

j + geom_step(direction = "hv"**)**
x, y, alpha, color, linetype, size

Visualizing error

df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
k <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))

k + geom_crossbar(fatten = 2**)**
x, y, ymax, ymin, alpha, color, fill, linetype, size

k + geom_errorbar()
x, ymax, ymin, alpha, color, linetype, size, width (also **geom_errorbarh()**)

k + geom_linerange()
x, ymin, ymax, alpha, color, linetype, size

k + geom_pointrange()
x, y, ymin, ymax, alpha, color, fill, linetype, shape, size

Maps

data <- data.frame(murder = USArrests$Murder,
state = tolower(rownames(USArrests)))
map <- map_data("state")
l <- ggplot(data, aes(fill = murder))

l + geom_map(aes(map_id = state), map = map**) +**
expand_limits(x = map$long, y = map$lat**)**
map_id, alpha, color, fill, linetype, size

Three Variables

seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2))
m <- ggplot(seals, aes(long, lat))

m + geom_contour(aes(z = z)**)**
x, y, z, alpha, colour, linetype, size, weight

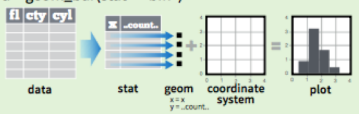
m + geom_raster(aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE**)**
x, y, alpha, fill

m + geom_tile(aes(fill = z)**)**
x, y, alpha, color, fill, linetype, size

GGPLOT2

Stats - An alternative way to build a layer

Some plots visualize a **transformation** of the original data set. Use a **stat** to choose a common transformation to visualize, e.g. a + geom_bar(stat = "bin")



Each stat creates additional variables to map aesthetics to. These variables use a common **..name..** syntax.

stat functions and geom functions both combine a stat with a geom to make a layer, i.e. **stat_bin(geom="bar")** does the same as **geom_bar(stat="bin")**

stat function | layer specific mappings | variable created by transformation

i + **stat_density2d**(aes(fill = ..level..), geom = "polygon", n = 100)

geom for layer | parameters for stat

1D distributions

a + **stat_bin**(binwidth = 1, origin = 10)
x, y | ..count.., ..ncount.., ..density.., ..ndensity..
a + **stat_bindot**(binwidth = 1, binaxis = "x")
x, y, | ..count.., ..ncount..
a + **stat_density**(adjust = 1, kernel = "gaussian")
x, y, | ..count.., ..density.., ..scaled..

2D distributions

f + **stat_bin2d**(bins = 30, drop = TRUE)
x, y, fill | ..count.., ..density..
f + **stat_binhex**(bins = 30)
x, y, fill | ..count.., ..density..
f + **stat_density2d**(contour = TRUE, n = 100)
x, y, color, size | ..level..

3 Variables

m + **stat_contour**(aes(z = z))
x, y, z, order | ..level..
m+ **stat_spoke**(aes(radius= z, angle = z))
angle, radius, x, xend, y, yend | ..x.., ..xend.., ..y.., ..yend..
m + **stat_summary_hex**(aes(z = z), bins = 30, fun = mean)
x, y, z, fill | ..value..
m + **stat_summary2d**(aes(z = z), bins = 30, fun = mean)
x, y, z, fill | ..value..

Comparisons

g + **stat_boxplot**(coef = 1.5)
x, y | ..lower.., ..middle.., ..upper.., ..outliers..
g + **stat_ydensity**(adjust = 1, kernel = "gaussian", scale = "area")
x, y | ..density.., ..scaled.., ..count.., ..n.., ..violinwidth.., ..width..

Functions

f + **stat_ecdf**(n = 40)
x, y | ..x.., ..y..
f + **stat_quantile**(quantiles = c(0.25, 0.5, 0.75), formula = y ~ log(x), method = "rq")
x, y | ..quantile.., ..x.., ..y..
f + **stat_smooth**(method = "auto", formula = y ~ x, se = TRUE, n = 80, fullrange = FALSE, level = 0.95)
x, y | ..se.., ..x.., ..y.., ..ymin.., ..ymax..

General Purpose

ggplot() + **stat_function**(aes(x = -3:3), fun = dnorm, n = 101, args = list(sd=0.5))
x | ..y..
f + **stat_identity**()
ggplot() + **stat_qq**(aes(sample=1:100), distribution = qt, dparams = list(df=5))
sample, x, y | ..x.., ..y..
f + **stat_sum**()
x, y, size | ..size..
f + **stat_summary**(fun.data = "mean_cl_boot")
f + **stat_unique**()

Scales

Scales control how a plot maps data values to the visual values of an aesthetic. To change the mapping, add a custom scale.

n <- b + **geom_bar**(aes(fill = fl))
n

scale_ | aesthetic to adjust | prepackaged scale to use | scale specific arguments

n + **scale_fill_manual**(
values = c("skyblue", "royalblue", "blue", "navy"),
limits = c("d", "e", "p", "r"), **breaks** =c("d", "e", "p", "r"),
name = "fuel", **labels** = c("D", "E", "P", "R"))

range of values to include in mapping | title to use in legend/axis | labels to use in legend/axis | breaks to use in legend/axis

General Purpose scales

Use with any aesthetic:
alpha, color, fill, linetype, shape, size

scale_*_continuous() - map cont' values to visual values
scale_*_discrete() - map discrete values to visual values
scale_*_identity() - use data values **as** visual values
scale_*_manual(values = c()) - map discrete values to manually chosen visual values

X and Y location scales

Use with x or y aesthetics (x shown here)

scale_x_date(labels = date_format("%m/%d"), breaks = date_breaks("2 weeks")) - treat x values as dates. See ?strptime for label formats.
scale_x_datetime() - treat x values as date times. Use same arguments as scale_x_date().
scale_x_log10() - Plot x on log10 scale
scale_x_reverse() - Reverse direction of x axis
scale_x_sqrt() - Plot x on square root scale

Color and fill scales

Discrete

n <- b + **geom_bar**(aes(fill = fl))
n + **scale_fill_brewer**(palette = "Blues")
For palette choices: library(RColorBrewer) display.brewer.all()
n + **scale_fill_grey**(start = 0.2, end = 0.8, na.value = "red")

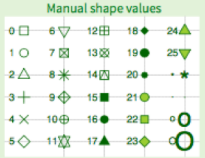
Continuous

o <- a + **geom_dotplot**(aes(fill = ..x..))
o + **scale_fill_gradient**(low = "red", high = "yellow")
o + **scale_fill_gradient2**(low = "red", hight = "blue", mid = "white", midpoint = 25)
o + **scale_fill_gradientn**(colours = terrain.colors(6))
Also: rainbow(), heat.colors(), topo.colors(), cm.colors(), RColorBrewer::brewer.pal()

Shape scales

p <- f + **geom_point**(aes(shape = fl))
p + **scale_shape**(solid = FALSE)
p + **scale_shape_manual**(values = c(3:7))
Shape values shown in chart on right

Manual shape values



Size scales

q <- f + **geom_point**(aes(size = cyl))
q + **scale_size_area**(max = 6)
Value mapped to area of circle (not radius)

Coordinate Systems

r <- b + geom_bar()

r + **coord_cartesian**(xlim = c(0, 5))
xlim, ylim
The default cartesian coordinate system

r + **coord_fixed**(ratio = 1/2)
ratio, xlim, ylim
Cartesian coordinates with fixed aspect ratio between x and y units

r + **coord_flip**()
xlim, ylim
Flipped Cartesian coordinates

r + **coord_polar**(theta = "x", direction=1)
theta, start, direction
Polar coordinates

r + **coord_trans**(ytrans = "sqrt")
xtrans, ytrans, limx, limy
Transformed cartesian coordinates. Set extras and strains to the name of a window function.

z + **coord_map**(projection = "ortho", orientation=c(41, -74, 0))
projection, orientation, xlim, ylim
Map projections from the mapproj package (mercator (default), azequalarea, lagrange, etc.)

Position Adjustments

Position adjustments determine how to arrange geoms that would otherwise occupy the same space.

s <- ggplot(mpg, aes(fl, fill = drv))

s + **geom_bar**(**position = "dodge"**)
Arrange elements side by side

s + **geom_bar**(**position = "fill"**)
Stack elements on top of one another, normalize height

s + **geom_bar**(**position = "stack"**)
Stack elements on top of one another

f + **geom_point**(**position = "jitter"**)
Add random noise to X and Y position of each element to avoid overplotting

Each position adjustment can be recast as a function with manual **width** and **height** arguments

s + **geom_bar**(**position = position_dodge(width = 1)**)

Themes

r + **theme_bw**()
White background with grid lines

r + **theme_classic**()
White background no gridlines

r + **theme_grey**()
Grey background (default theme)

r + **theme_minimal**()
Minimal theme

ggthemes - Package with additional ggplot2 themes

Faceting

Facets divide a plot into subplots based on the values of one or more discrete variables.

t <- ggplot(mpg, aes(cty, hwy)) + geom_point()

t + **facet_grid**(. ~ fl)
facet into columns based on fl

t + **facet_grid**(year ~ .)
facet into rows based on year

t + **facet_grid**(year ~ fl)
facet into both rows and columns

t + **facet_wrap**(~ fl)
wrap facets into a rectangular layout

Set **scales** to let axis limits vary across facets

t + **facet_grid**(y ~ x, **scales = "free"**)
x and y axis limits adjust to individual facets
- **"free_x"** - x axis limits adjust
- **"free_y"** - y axis limits adjust

Set **labeller** to adjust facet labels

t + **facet_grid**(. ~ fl, **labeller = label_both**)

fl: c	fl: d	fl: e	fl: p	fl: r

t + **facet_grid**(. ~ fl, **labeller = label_bquote**(alpha ^ .(x)))

α^{c}	α^{d}	α^{e}	α^{p}	α^{r}

t + **facet_grid**(. ~ fl, **labeller = label_parsed**)

c	d	e	p	r

Labels

t + **ggtitle**("New Plot Title")
Add a main title above the plot
t + **xlab**("New X label")
Change the label on the X axis
t + **ylab**("New Y label")
Change the label on the Y axis
t + **labs**(title ="New title", x = "New x", y = "New y")
All of the above

Use scale functions to update legend labels

Legends

t + **theme**(legend.position = "bottom")
Place legend at "bottom", "top", "left", or "right"
t + **guides**(color = "none")
Set legend type for each aesthetic: colorbar, legend, or none (no legend)
t + **scale_fill_discrete**(name = "Title", labels = c("A", "B", "C"))
Set legend title and labels with a scale function.

Zooming

Without clipping (preferred)

t + **coord_cartesian**(
xlim = c(0, 100), ylim = c(10, 20))

With clipping (removes unseen data points)

t + **xlim**(0, 100) + **ylim**(10, 20)
t + **scale_x_continuous**(limits = c(0, 100)) + **scale_y_continuous**(limits = c(0, 100))